Parallel Architectures and Adaptation Algorithms for Programmable FIR Digital Filters With Fully Pipelined Data and Control Flows

BEHROOZ PARHAMI AND DING-MING KWAI[*]
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560, U.S.A.
E-mail: parhami@ece.ucsb.edu
[*]*Worldwide Semiconductor Corporation*
Hsinchu, 300 Taiwan

Previous designs of programmable finite impulse response (FIR) digital filters have demonstrated that the use of broadcast input data and control can lead to a high performance-to-cost ratio. As the technology moves into deeper submicron regimes, this approach should be reexamined by paying greater attention to the effect of interconnects. In this paper, we quantify the contribution of interconnect delay to the cycle time and demonstrate its negative effects on both scalability and cost-effectiveness of such broadcast designs. We further show how speed and density improvements secured through technology scaling can be maintained by a fully pipelined design in which both data and control signals are restricted to local connections. One important feature of our design is that the data input port is reused for delivering the new coefficients. Consequently, coefficients can be loaded in bit-parallel form with no increase in the number of input pins, thereby facilitating and speeding up run-time adaptation to the application environment. Another feature is that variable-precision coefficients can be accommodated easily and flexibly, with no speed penalty. Because the inner-product computation at the heart of a FIR filter occurs in many other signal processing applications, our design methods and conclusions are widely applicable to the design of application-specific and embedded parallel architectures.

Keywords: application-specific parallel architecture, data-driven control, embedded system, pipelining, reconfigurable architecture, speed-cost tradeoffs, technology scaling

1. INTRODUCTION

In view of their simplicity, finite impulse response (FIR) digital filters are used in a variety of practical signal processing applications, particularly those requiring inexpensive hardware implementations for embedded systems. A FIR digital filter essentially computes the inner product of a data vector and a coefficient (weight) vector. Simple FIR filters have their coefficients built into the circuit or burned into read-only memory. In some applications, flexibility and adaptation requirements dictate that the coefficients

Received September 28, 2001; accepted April 15, 2002.
Communicated by Jang-Ping Sheu, Makoto Takizawa and Myongsoon Park.

be changeable in the field or be dynamically modifiable at set-up or run time. Such filters are said to be programmable. Programming of such digital filters involves loading a coefficient memory with new values or shifting in the new coefficients into special registers, with the latter approach preferred when high performance or quicker adaptation is needed.

Conventional designs for programmable FIR digital filters are based on the transposed direct form realization [21]. In this scheme, an *N*-tap FIR filter is implemented as a cascade architecture consisting of *N* fixed-size cells, one for each tap. Input data are broadcast to all filter taps and their output responses computed as inner-product steps in pipelined fashion [7, 10-13, 18, 25, 28]. Coefficients are loaded in either bit-serial or bit-parallel form, by connecting all coefficient registers into a chain whose shifting is activated by a common enable line. Designs opting for bit-serial loading of the coefficients reduce the number of pins but are only suitable for application contexts involving slow adaptation, given the long programming time [10].

As the required number of taps may vary from one application to another, designs must be modular and readily expandable. Tap counts from 8 to 64 are found in different applications [7, 10, 11, 13, 18, 25, 28]. In adaptive applications, an “enable” broadcast signal is used in connection with coefficient modification. When this signal is deasserted, desired coefficient values are input through the pipeline registers, which form a shift-register chain. When the enable signal is asserted, the contents of the pipeline registers are simultaneously saved as new coefficient values. The choice of a broadcast control scheme, as a simple extension of the input data bus, makes the control broadcast overhead relatively insignificant, in view of the latter already being in place, leading to cost-effective designs.

While the preceding assessment may be accurate for short filters, the reduced depth of pipeline stages heightens the significance of propagation delays on wires [1, 2, 15]. For a long filter, the broadcast overhead for input signals becomes a major portion of the cycle time. The cycle time is affected by two factors: broadcasting data to all filter taps and performing a pipelined operation on data in each tap. The required cycle time increases with *N* simply because longer interconnections will be involved and a larger load must be driven.

In this paper, we propose a fully pipelined design in which both data and control signals are pipelined through the filter, thus restricting the global connection to clock and power supply. Shorter interconnects allow the use of a faster clock rate that is independent of the filter size; so even though the *N*-tap filter’s latency is *N* clock cycles for both the broadcast and pipelined designs, the latter may be significantly faster. It is our goal to quantify the differences in speed and relate them to the overall cost-effectiveness of the various designs. Such a study needs realistic analyses based on an architecture- and technology-dependent model that exposes the tradeoffs involved.

Our design is completely modular and allows the corresponding implementations to take full advantage of the improved raw circuit performance due to the scaling of dimensions in integrated circuits. We note that the broadcast overhead cannot be alleviated by merely pipelining the data signals, because the overhead will then shift to the control signals. Pipelining both data and control signals appears to be the only viable alternative. Using data from 1.0 and 0.5 μm CMOS technologies, we show that the increased area to accommodate more pipeline registers can be compensated for by the higher operating frequency, thus maintaining the throughput rate with scaling. One important feature of our design is that the data input port is reused for delivering the new coefficients. Consequently, coefficients can be loaded in bit-parallel form with no increase in the number of input pins, thereby facilitating and speeding up run-time adaptation to the application environment. Another feature is that variable-precision coefficients can be accommodated easily and flexibly, with virtually no speed penalty.

Our presentation in the rest of this paper is organized as follows. In section 2, we categorize four modes of data and control flow that can be used in designing programmable FIR digital filters and also derive our fully pipelined design. Section 3 discusses the cost-performance tradeoffs and the effects of scaling on the above designs. In section 4, we show that our fully pipelined design allows us to extend the precision of coefficients without degrading the speed, while a similarly flexible broadcast design would imply an elongated cycle time. Section 5 contains our conclusions.

2. FULLY PIPELINED FIR FILTERS

2.1 Modes of Data and Control Flow

The fully pipelined design is derived from the direct form by successively applying retiming transformations [8, 19]. Fig. 1 shows the retimed realization of a FIR filter of order N. The same structure can also be derived by using the dependence method [14]. We refer the reader to [6] for other variants.

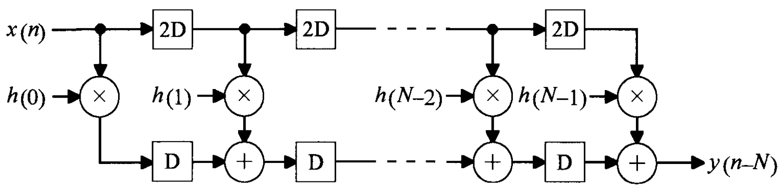


Fig. 1. Retimed direct-form realization of an FIR filter of order N.

For a programmable design, mechanisms must be provided to input and store the coefficients. Given that the data and control signals can be either broadcast to, or pipelined through, the filter, four categories of design with regard to data and control flow can be distinguished.

1. Broadcast control, broadcast data (BCBD): Input data are broadcast to all filter taps and the coefficients are pipelined through the registers via a separate input port. The common “enable” control signal is asserted when the registers are to be filled up with new values for the coefficients. These coefficients are input in order from $h(0)$ to $h(N - 1)$; see Fig. 2a.
2. Pipelined control, broadcast data (PCBD): The coefficients share the same input port with broadcast data. The control signal is shifted through the filter, sequentially enabling the coefficient registers to store their new values in order from $h(N - 1)$ to $h(0)$; see Fig. 2b.

3. Broadcast control, pipelined data (BCPD): Input data and coefficients are pipelined through all taps using two separate input ports. The common control signal is asserted until the registers are filled up with the coefficients, which are input in order from $h(0)$ to $h(N-1)$; see Fig. 2c.
4. Pipelined control, pipelined data (PCPD): Coefficients share the same input port with pipelined data. If the coefficients are input from $h(N-1)$ to $h(0)$ before the assertion of control and input of data, the pipelined control signal arrives at a tap just when its coefficient reaches the register; see Fig. 2d.

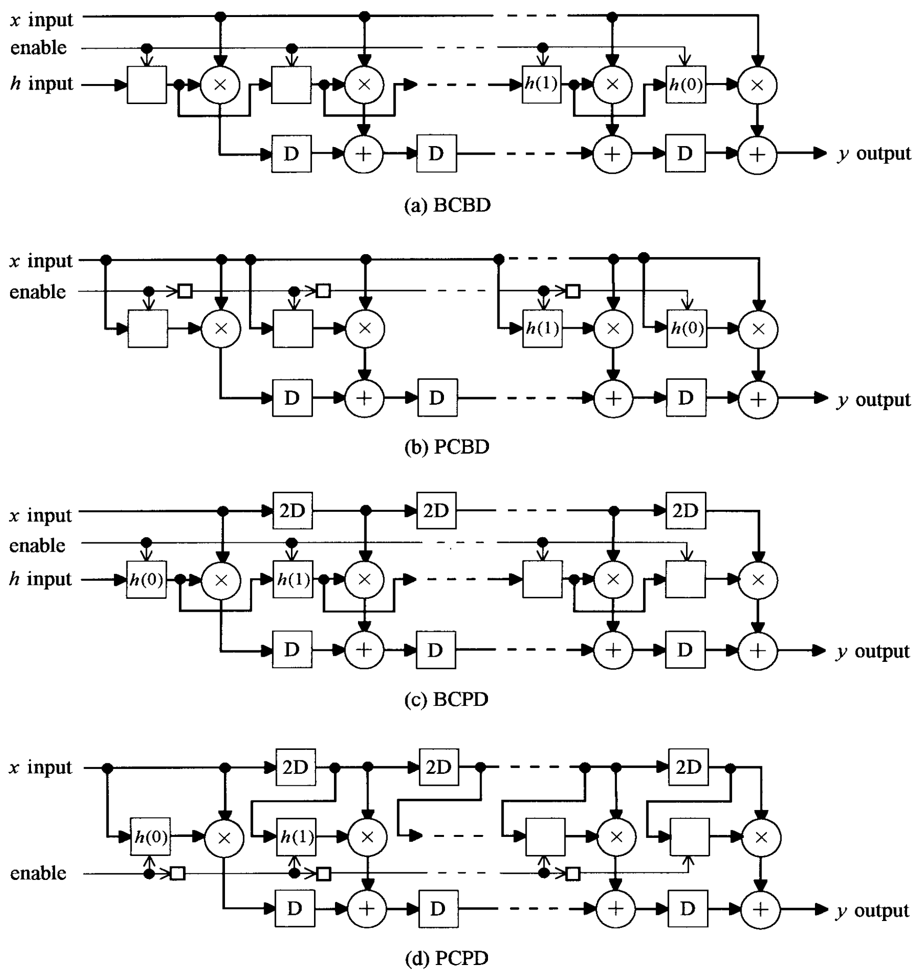


Fig. 2. The four modes of data and control flow in programmable FIR digital filters: (a) broadcast control, broadcast data, (b) pipelined control, broadcast data, (c) broadcast control, pipelined data, and (d) pipelined control, pipelined data.

Although the PCBD mode has the advantage that the coefficients can be loaded in bit-parallel form without increasing the number of pins, this sharing of data input port can seriously aggravate the data broadcast overhead. Most existing designs ignore the control broadcast overhead and adopt the BCBD mode. Their focus, instead, is on optimizing the multiply-add operation which, in effect, leads to the reduction of circuit depth within each cell. Ironically, the speed gained by such reductions in circuit depth exacerbates the effect of propagation delays on long wires.

We note that even if the BCPD mode is used, broadcast overhead is not totally eliminated, because we still need to broadcast the control signal. If reprogramming is rather infrequent, it is possible to design the filter such that the overhead is not paid during normal operation. This, however, leads to added circuit complexity and may degrade the cost-effectiveness of the resulting design.

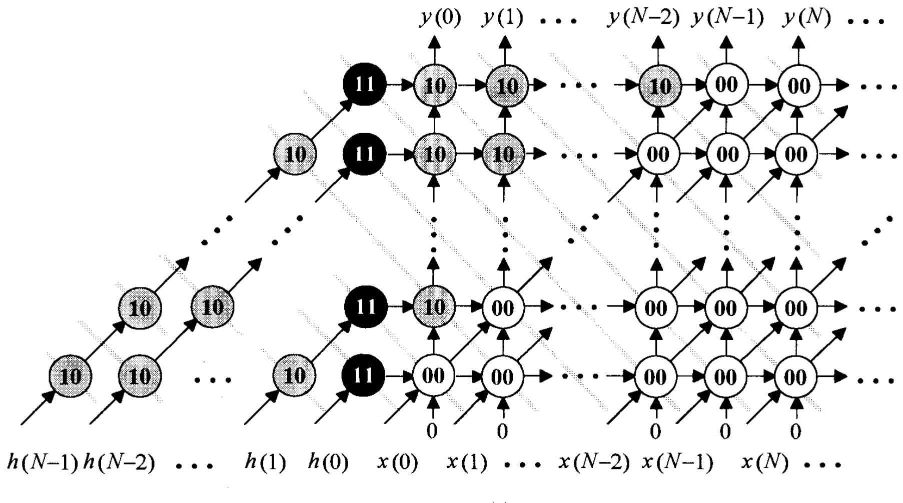
2.2 Deriving the Pipelined Control

We now derive the pipelined control signals in the PCPD mode, which we refer to as a fully pipelined design. The pipelining of control signals with the data signals can be interpreted as attaching special control tags to the data streams $\boldsymbol{X} = \{x(0), x(1), x(2), \ldots\}$ and $\boldsymbol{Y} = \{y(0), y(1), y(2), \ldots\}$. The coefficients are loaded, via the x input port, in the descending order $h(N-1), h(N-2), \ldots, h(0)$, prior to the insertion of the first input data item $x(0)$. The loading elongates the data stream $\boldsymbol{X}$ by N and takes N clock cycles for any subsequent change.

To facilitate our derivation, we unroll the signal flow graph of Fig. 1 along time steps into a dependence graph, as depicted in Fig. 3. The original signal flow graph can then be seen as a projection of the dependence graph along the horizontal direction. The light diagonal lines show the schedule. The dependence graph is augmented with nodes acting to pass and store the coefficients (black nodes, and shaded nodes to their left). Once the coefficient $h(i)$ turns into the projection (horizontal) direction, it will be stored at the tap. The shaded nodes to the right of the black nodes are needed because we have to ensure $x(i) = 0$ for $i < 0$ in order to apply the formula $y(n) = \sum_{i=0,n} h(i)\, x(n-i)$ uniformly. Thus, $y(n)$, $0 \le n \le N-2$, simply passes through taps $h(n+1), \ldots, h(N-1)$ with no change.

The nodes in Fig. 3 have been labeled with binary x and y tags that together specify the node function: 11 for store, 10 for pass, and 00 for multiply-add operation. The fourth combination 01 is not used in this design. The preceding tag assignment can be derived as follows. We observe that the nodes for multiply-add operations are aligned along the diagonal direction of the data stream $\boldsymbol{X}$. Hence, we assign an x tag of 0 to distinguish these nodes from the other two node types; i.e., pass and store. Similarly, the nodes for store are aligned along the vertical direction, which corresponds to the flow direction of the data stream $\boldsymbol{Y}$. Hence, we assign a y tag of 1 to distinguish these store nodes from the other two node types; namely, pass and multiply-add. The combination of the x and y tags, when data streams $\boldsymbol{X}$ and $\boldsymbol{Y}$ meet at a node, results in the assignments depicted in Fig. 3b [17].

The coding scheme used for the tags corresponds to the following simple interpretations of the control signals. An x tag of 1 indicates that the associated input data item is a coefficient, while a y tag of 1 distinguishes the input data element as the last item in the sequence of coefficients.


y(0) y(1) ... y(N–2) y(N–1) y(N) ...
h(N–1) h(N–2) ... h(1) h(0) x(0) x(1) ... x(N–2) x(N–1) x(N) ...
(a)

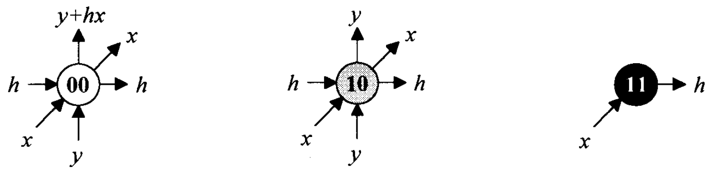


Fig. 3. (a) Dependence graph of a programmable FIR filter, featuring three node types. (b) Node definitions.

2.3 Basic Cell Structure

Based on the coding derived in section 2.2, the basic cell structure for the programmable FIR filter can be easily obtained (Fig. 4). The added ***Z*** data stream allows us to efficiently realize a linear-phase FIR filter of order $2N$ (or $2N - 1$) by the well-known method of folding it into N taps, taking advantage of the symmetric property of its coefficients $h(i) = h(2N - 1 - i)$, $0 \le i \le n - 1$. To see this, rewrite the system function as

$$\begin{aligned} y(n) &= \textstyle\sum_{i=0,2N-1} h(i)\, x(n-i) \\ &= \textstyle\sum_{i=0,N-1} h(i)\, x(n-i) + \sum_{i=0,N-1} h(i)\, x(n+i-2N+1) \end{aligned}$$

and rename the second summation term in the last expression as $z(n - 2N + 2i + 1)$, where $z(n) = \sum_{i=0,N-1} h(i)\, x(n - i)$. Thus, the computation of $z(n - 2N + 2i + 1)$ is the same as that of $y(n - 2N + 2i + 1)$ by the first N taps. The index difference $2N - 2i - 1$ between $y(n)$ and $z(n - 2N + 2i + 1)$ implies a delayed y output with two more pipeline registers inserted between cells and one register added to the z output following the last cell. Fig. 5 shows the first and last cells of the linear-phase FIR filter of even order $2N$. In a similar manner, the folded linear-phase FIR filter of odd order $2N - 1$ can be derived by removing the register added to the z output following the last cell.

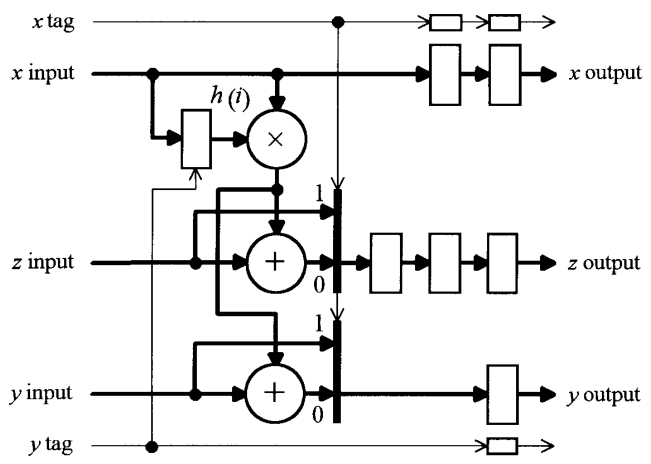


Fig. 4. The basic cell structure of the programmable FIR filter with pipelined data and control.

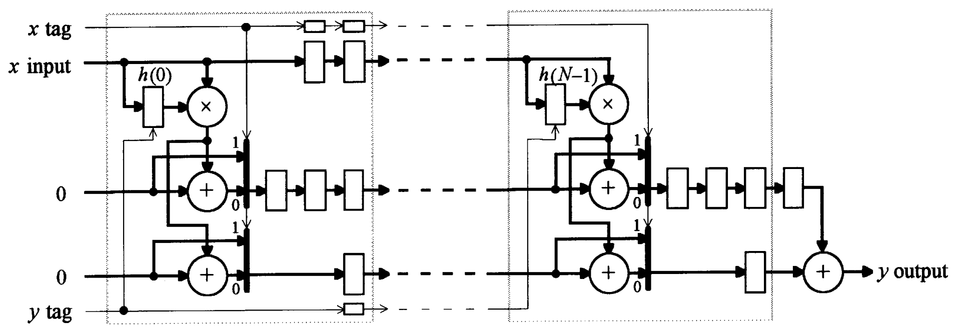


Fig. 5. The first and last cells of a programmable linear-phase FIR filter of order 2*N*.

3. TRADEOFFS AND SCALABILITY

In this section, we compare the various filter designs with respect to cost-performance ratio and also examine their scalability. Only BCBD and PCPD modes are considered, which represent the two extremes with lowest cost and highest performance, respectively. To facilitate our analyses, we assume data to be represented by fixed-point fractional numbers with 16-bit inputs, 16-bit coefficients, and 32-bit outputs. Our concern here is the core array. When implemented on a single chip, the output is often truncated or rounded and extra hardware is required on the periphery.

The coefficients are assumed to be encoded using a canonic signed-digit (CSD) representation, using a minimal number of nonzero digits from the digit set {–1, 0, 1}. This encoding, which reduces the number of partial products to be generated in view of the absence of consecutive nonzero digits, is standard. It has been used in several FIR filter designs as well as some automatic synthesis tools [7, 11-13, 18, 28]. For simplicity, let us select two signed digits to represent each coefficient value:

$$h(i) = s_0(i)\, 2^{-p_0(i)} + s_1(i)\, 2^{-p_1(i)}$$

The coefficient $h(i)$ is fully characterized by the four-tuple $(s_0(i), p_0(i), s_1(i), p_1(i))$, where $s_0(i), s_1(i) \in \{-1, 0, 1\}$, $p_0(i) \in \{0, 1, \ldots, 13\}$, and $p_1(i) \in \{2, 3, \ldots, 15\}$, thus requiring $2(2 + 4) = 12$ bits for its representation [20].

The two partial products are generated by shifting the input data $x(n - i)$ by $p_0(i)$ and $p_1(i)$ positions, respectively, and complementing if required. The two partial products are added with the sum and carry from the previous cell by a two-level carry-save adder tree. However, to avoid carry propagation delay in each cell, the true result is not computed until after the last cell. This technique too, which replaces multiplications by carry-free additions, is common practice in high-performance filter design [11].

3.1 Comparison for 1.0 μm CMOS

We estimate the area and cycle time for our filter designs using a 1.0 μm CMOS technology. Compared to the broadcast (BCBD) design, the layout of the fully pipelined (PCPD) design shows about 20% increase in area for each cell. Fig. 6 shows the corresponding relative cycle times and area-time products for different values of N. The area-time product, obtained by multiplying the relative area and cycle time, is a commonly used measure of cost-effectiveness.

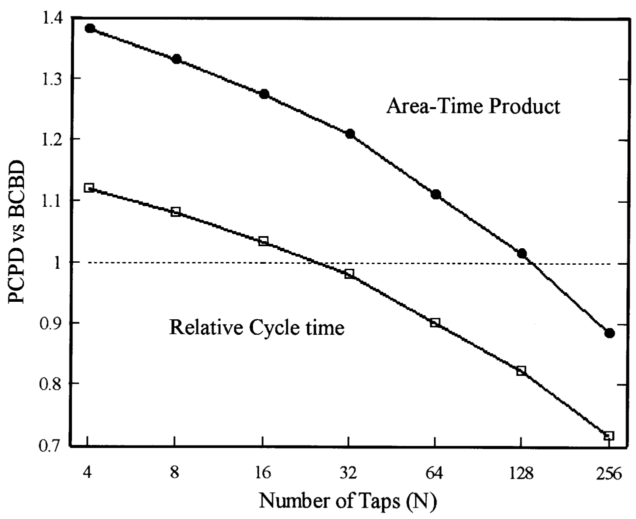


Fig. 6. Cycle time and area-time product of the fully pipelined design relative to the broadcast design, using 1.0 μm CMOS technology.

According to Fig. 6, the speed improvement resulting from pipelined data and control signals is not enough to offset the area increase for $N \leq 128$. This is because the original cell has a relatively low circuit depth and by adding control logic to the critical path, our design increases the cycle time for small N. Thus, for existing applications with $N \leq 64$, the broadcast design is more cost-effective when relatively old implementation technologies are used. In such cases, signal propagation delays incurred on long wires do not significantly affect the cycle time, making the broadcast design unconditionally better for small N (say, $N \leq 16$), while offering a worthwhile speed-for-area tradeoff in the case of larger N (say, $32 \leq N \leq 64$).

3.2 Comparison for 0.5 μm CMOS

The results of section 3.1 lose their validity with continued downward scaling of feature size for devices and wires in integrated circuits. As the dimensions are scaled down by a given factor, wire lengths are expected to shrink by the same factor. However, signal propagation delays on wires are not reduced at the same rate as device switching times [3, 4, 22]. The interconnect delay is a function of the wire resistance and capacitance (RC delay). Whereas wire resistance increases due to a reduction in cross-sectional area, there is no corresponding decrease in the wire capacitance, leading to a net increase in interconnect delay.

We thus need to reexamine the relative measures of layout area and cycle time for different values of N, with each change in technology. Fig. 7 shows the results for 0.5 μm CMOS technology. Compared to the results for 1.0-micron CMOS, the relative cycle time is now seen to drop more rapidly. While the observation that downward technology scaling has adverse effects on signal propagation delays is not new, there is a dearth of published work on quantifying the degradation and analyzing the tradeoffs.

According to Fig. 7, the area saved by the broadcast design is not enough to offset the speed degradation as N is extended beyond 16. An alternate view of our simulation results appears in Fig. 8 which shows the speed improvement due to the scaling down of the feature size from 1.0 to 0.5 μm. The superiority of the fully pipelined design in preserving the speed and density benefits of scaling is clearly visible. Even cautious interpolation of these results to the current leading edge and future technologies points to the inevitability of fully pipelined design for high-performance FIR digital filters of even modest size (say, $N \leq 16$).

Fig. 9 [16] shows the interconnect delay on wires for 1.0, 0.5, and 0.25 μm CMOS, presented as ratios with respect to the device switching time for the corresponding technology [3, 4, 9]. For the broadcast design, we have conservatively estimated the wire length in the range of 2 mm (8 taps) to 6 mm (64 taps), using a 1.0 μm CMOS technology and distributing the broadcast data by a standard treelike network [26]. These base points appear on the bottom curve in Fig. 9.

4. MORE PRECISE COEFFICIENTS

In section 3, we assumed that each coefficient is represented by two signed digits and their corresponding positions (powers of two). In practical applications, some coefficients may need more precision to satisfy given criteria [24, 27]. To maintain the modularity, we can decompose a more precise coefficient

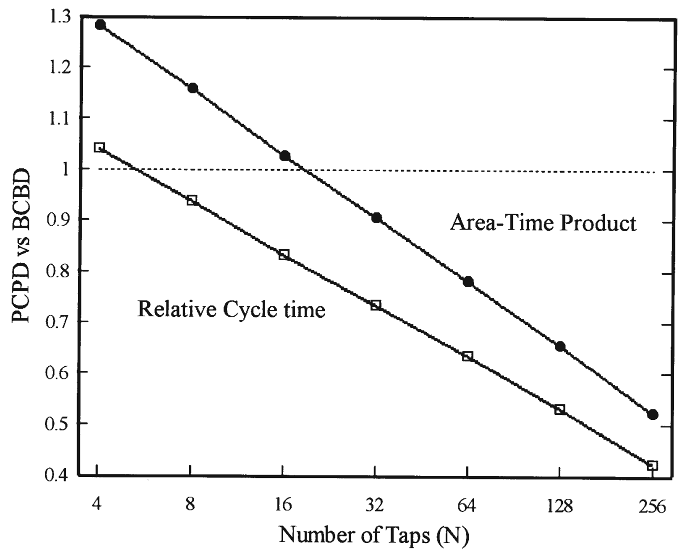


Fig. 7. Cycle time and area-time product of the fully pipelined design relative to the broadcast design, using 0.5 μm CMOS technology.

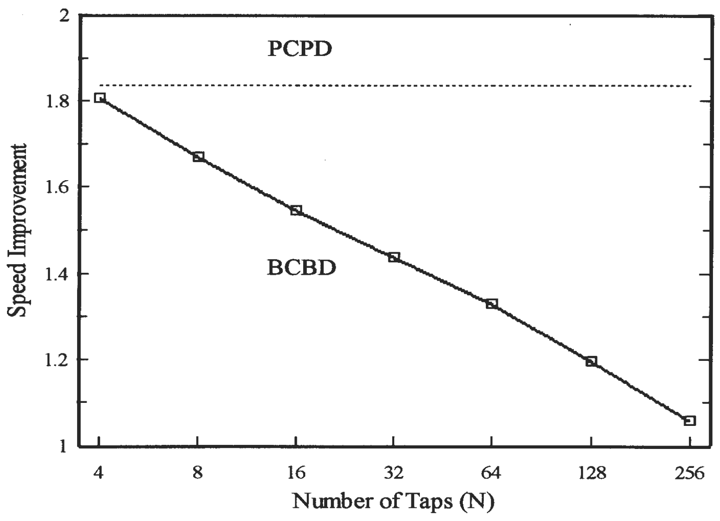


Fig. 8. The speed improvement of fully pipelined (PCPD) and broadcast (BCBD) designs when the feature size is scaled from 1.0 to 0.5 μm.

$$h(i) = \sum_{k=0,L(i)-1} s_k(i)\, 2^{-p_k(i)}$$

with $L(i)$ signed digits into consecutive two-digit pairs $h_j(i)$, $0 \le j \le \lceil L(i)/2 \rceil - 1$, such that

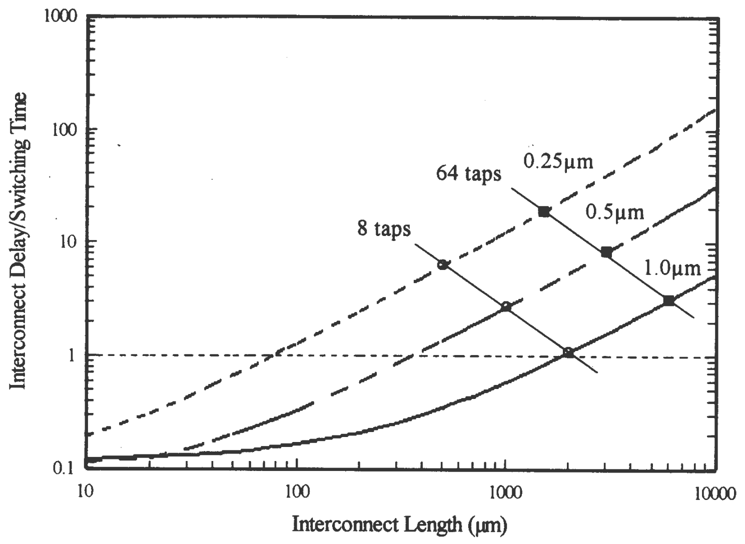


Fig. 9. The ratio of interconnect delay to device switching time versus wire length for 1.0, 0.5, and 0.25 μm CMOS technologies using standard aluminum/silicon dioxide-based metallization.

$$h(i) = \textstyle\sum_{j=0,\lceil L(i)/2\rceil-1} h_j(i)\, 2^{-p_k^{(i)}} = \sum_{j=0,\lceil L(i)/2\rceil-1} (s_{2j}(i)\, 2^{-p_{2j}^{(i)}} + s_{2j+1}(i)\, 2^{-p_{2j+1}^{(i)}})$$

In the broadcast design, the partial products $h_j(i)\ x(n-i)$ are accumulated in such a way that each of the y outputs of the first $\lceil L(i)/2\rceil - 1$ cells is passed directly to the next cell without going through a pipeline register [13]. This is akin to using a $2\lceil L(i)/2\rceil$-level carry-save adder tree to reduce $2\lceil L(i)/2\rceil + 2$ operands to 2 operands. However, because the broadcast input $x(n-i)$ lasts only for one clock cycle on the data bus, completing such a computation necessitates a significant increase in cycle time. As a result, extension of precision for the coefficients implies a serious degradation in speed.

In the fully pipelined design, such a flexibility can be provided with virtually no sacrifice in speed. We note that in Fig. 1, the data item $x(n-i)$ will lag one clock cycle behind $y(n)$ once they meet at cell $h_j(i)$. Thus, to let them meet again at the next cell $h_{j+1}(i)$, we can simply allow $x(n-i)$ to skip over one pipeline register, thereby reducing its 2D delay to D. Fig. 10 shows an example with $N = 3$, $L(0) = L(2) = 2$, and $L(1) = 4$. It is clear that the pipelined data flow mode is preserved.

Programming the delay of the x output requires a control signal that enables or disables the transmission gate added to bypass the register [13]. Any change in the precision of coefficients must be preceded by resetting of the delays, so that the pipelined control flow can effect the subsequent pass and store operations. This can be conveniently accomplished by using the unassigned tag value 01 to reset the delay. When multiply-add operations are performed, all x tags are labeled with 0s. We insert a y tag of 1 before the change. The combination of the y tag of 1 with previously inserted x tags of 0 yields a sequence of pipelined resets. Fig. 11 shows the dependence graph corresponding to the programmable FIR filter of Fig. 10. Heavy arrows indicate the flow of data stream $\boldsymbol{X}$.

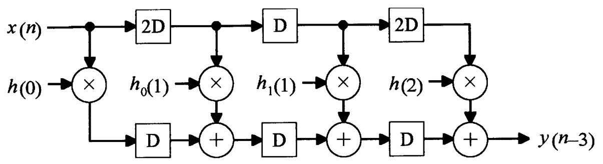


Fig. 10. An FIR digital filter of order 3, with the coefficient $h(1)$ having 4 digits in its CSD representation.

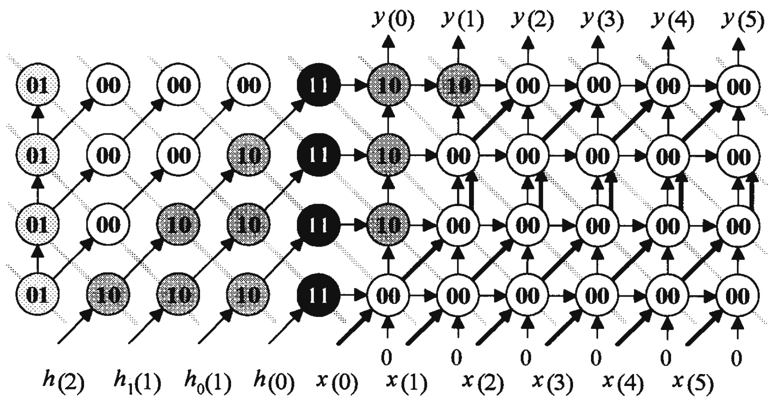

y(0) y(1) y(2) y(3) y(4) y(5)
h(2) h1(1) h0(1) h(0) x(0) x(1) x(2) x(3) x(4) x(5)

Fig. 11. The dependence graph corresponding to the programmable FIR filter of Fig. 10.

With the CSD representation, the width of the data input port is likely not used in its entirety for delivering the coefficients. Therefore, it can easily accommodate the single-bit bypass flag with each coefficient to indicate its continuation. The coefficient register in each cell must also be expanded by one bit. Fig. 12 shows the resulting cell structure. For simplicity, we have omitted the linear-phase part which is not affected by the extended-precision modification.

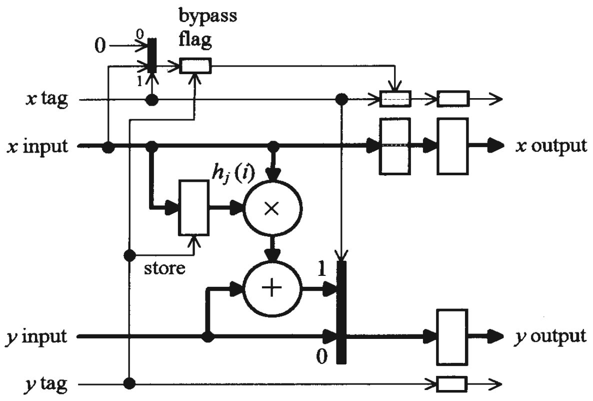


Fig. 12. The basic cell structure allowing more precise coefficients to span multiple cells.

The pipelined mode of control, using the *x* and *y* tags, is as follows. When the *x* tag is 0 and the *y* tag is 1, the multiplexer at the top of Fig. 12 selects 0 to be stored as the bypass flag, thus disabling the bypass. The coefficients, received with *x* tag of 1 and *y* tag of 0, can then be loaded through the registers until the *y* tag becomes 1. Then, the coefficients are stored and the bypass path is set up. Multiply-add operations are performed on the subsequently inserted data items carrying *x* tag of 0 and *y* tag of 0.

5. CONCLUSIONS

Different designs, involving broadcast of data or control signals, have been proposed for the realization of programmable FIR filters. Our results show that whereas the broadcast design leads to a high cost-performance ratio with the 1.0 µm CMOS technology, especially when the filter size is small, more advanced implementation technologies or larger filters lead to severe overheads. This is due to the contribution of signal propagation on long wires to the clock cycle time.

To alleviate the broadcasting overhead, we proposed a fully pipelined design that restricts the global connections to clock and power supply. We showed that such a fully pipelined design is feasible and requires only one control tag bit per data element. Our data-driven control scheme thus requires two tag bits in all (compared to one control bit in the broadcast design) to effect the desired cell operations. To deliver the coefficients into the filter, the broadcast design requires a separate coefficient input port. Our fully pipelined design, by contrast, reuses the data input port for this purpose. Additionally, we showed that the precision of coefficients can be extended easily and flexibly in our design, with no sacrifice in speed.

Even though continued downward scaling of feature size will make the fully pipelined design a necessity in future, the current density levels may favor a hybrid design in which pipeline stages are inserted after every *k*-tap segment and broadcasting is used within each segment. We have analyzed the resulting hybrid architectures in a companion paper [16], concluding that by restricting the segment size to a small number of taps, the benefits of scaling can be secured. Our results serve to reiterate the importance of local and pipelined communication in modular designs, based on state-of-the-art VLSI technology.

Modifications to reduce the wire resistance and capacitance through the use of new conductor (e.g., copper) and insulator (e.g., polymer) materials, as well as multi-level interconnections, do not significantly affect our conclusions. Optimistically, the interconnect RC time constant can be reduced by a factor of three [5]. Such a change is equivalent to dividing the interconnect delay/switching time ratio of Fig. 9 by three, and thus shifting the optimal segment size by the same factor. As such a change involves greater development and production complexities, the resulting cost factors must be taken into account for a fair and complete comparison.

Our data-driven control scheme is quite general [17] and can be applied to the design of many embedded or application-specific systems that require extremely high performance. Notable among such applications is the design of an insertion sorter [23] that doubles as a priority queue and can handle insertions and extractions with no idle clock cycle in between consecutive operations of either type. Other applications are under investigation, as are error detection and fault tolerance schemes.

ACKNOWLEDGMENT

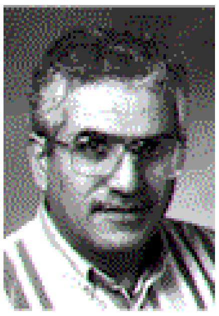
We thank Dr. Chih-Chuan Lin at SGS-Thompson for his help with the technology aspects of this paper.

REFERENCES

1. M. Afghahi and C. Svensson, “Performance of synchronous and asynchronous schemes for VLSI systems,” *IEEE Transactions on Computers*, Vol. 41, 1992, pp. 858-872.
2. D. Audet, Y. Savaria, and N. Arel, “Pipelined communications in large VLSI/ULSI systems,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, 1994, pp. 1-10.
3. S. Bothra, B. Rogers, M. Kellam, and C.M. Osburn, “Analysis of the effects of scaling on interconnect delay in ULSI circuits,” *IEEE Transactions on Electron Devices*, Vol. 40, 1993, pp. 591-597.
4. B. Davari, R. H. Dennard, and G. G. Shahidi, “CMOS scaling for high performance and low power – the next ten years,” in *Proceedings of the IEEE*, Vol. 83, 1995, pp. 595-606.
5. D. C. Edelstein, G. A. Sai-Halasz, and Y.-J. Mii, “VLSI on-chip interconnection performance simulations and measurements,” *IBM Journal of Research and Development*, Vol. 39, 1995, pp. 383-401.
6. D. J. Evans and M. Gusev, “New linear systolic arrays for digital filters and convolution,” *Parallel Computing*, Vol. 20, 1994, pp. 29-61.
7. J. B. Evans, “Efficient FIR filter architectures suitable for FPGA implementation,” *IEEE Transactions on Circuits and Systems II*, Vol. 41, 1994, pp. 490-493.
8. G. Even, “The retiming lemma: a simple proof and applications,” *Integration: The VLSI Journal*, Vol. 20, 1996, pp. 123-137.
9. M. Gilligan and S. Gupta, “A methodology for estimating interconnect capacitance for signal propagation delay in VLSIs,” *Microelectronics Journal*, Vol. 26, 1995, pp. 327-336.
10. M. Hatamian and S. K. Rao, “A 100 MHz 40-tap programmable FIR filter chip,” in *Proceedings of the International Symposium on Circuits and Systems*, 1990, pp. 3053-3056.
11. R. A. Hawley et al., “Design techniques for silicon compiler implementations of high-speed FIR digital filters,” *IEEE Journal of Solid-State Circuits*, Vol. 31, 1996, pp. 656-667.
12. R. Jain, P. T. Yang, and T. Yoshino, “FIRGEN: a computer-aided design system for high performance FIR filter integrated circuits,” *IEEE Transactions on Signal Processing*, Vol. 39, 1991, pp. 1655-1668.
13. K. Y. Khoo, A. Kwentus, and A. N. Wilson, “A programmable FIR digital filter using CSD coefficients,” *IEEE Journal of Solid-State Circuits*, Vol. 31, 1996, pp. 869-864.
14. S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
15. D.-M. Kwai and B. Parhami, “Area-time tradeoffs in FIR digital filters with broad-

cast and pipelined designs," in *Proceedings of the Midwest Symposium on Circuits and Systems*, Vol. 1, 1997, pp. 449-452.

16. D.-M. Kwai and B. Parhami, "Scalability of programmable FIR digital filters," *Journal of VLSI Signal Processing*, Vol. 21, 1999, pp. 31-35.
17. D.-M. Kwai and B. Parhami, "High-performance array processing with fully pipelined data streams and control paths," in *Proceedings of the International Conference on Parallel and Distributed Computing and Systems*, 1999, pp. 609-612.
18. J. Laskowski and H. Samueli, "A 150-MHz 43-tap half-band FIR digital filter in 1.2-m CMOS generated by silicon compiler," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1992, pp. 11.4.1-11.4.4.
19. C. E. Leiserson and J. B. Saxe, "Optimizing synchronous systems," *Journal of VLSI Computer Systems*, Vol. 1, 1983, pp. 41-67.
20. W. J. Oh and Y. H. Lee, "Implementation of programmable multiplierless FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems II*, Vol. 42, 1995, pp. 553-556.
21. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
22. B. Parhami, *Introduction to Parallel Processing: Algorithms and Architectures*, Plenum Press, New York, 1999.
23. B. Parhami and D.-M. Kwai, "Data-driven control scheme for linear arrays: application to a stable insertion sorter," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, 1999, pp. 23-28.
24. H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, Vol. 36, 1989, pp. 1044-1047.
25. L. E. Thon, P. Sutardja, F.-S. Lai, and G. Coleman, "A 240 MHz 8-tap programmable FIR filter for disk-drive read channels," *Digest of the IEEE International Solid-State Circuits Conference*, 1995, pp. 82-83.
26. C. Y. Wu and M.-C. Shiau, "Delay models and speed improvement techniques for RC tree interconnections among small geometry CMOS inverters," *IEEE Journal of Solid-State Circuits*, Vol. 25, 1990, pp. 1247-1256.
27. M. Yagyu, A. Nishihara, and N. Fujii, "Design of FIR digital filters using estimates of error function over CSD coefficient space," *IEICE Transactions Fundamentals*, Vol. E79-A, 1996, pp. 283-290.
28. T. Yoshino et al., "A 100-MHz 64-tap FIR digital filter in 0.8-m BiCMOS gate array," *IEEE Journal of Solid-State Circuits*, Vol. 25, 1990, pp. 1494-1501.



Behrooz Parhami received his Ph.D. in computer science from University of California, Los Angeles, in 1973. Presently, he is Professor in the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His research deals with parallel architectures and algorithms, computer arithmetic, and reliable computing. In his previous position with Sharif University of Technology in Tehran, Iran (1974-88), he was also involved in the areas of educational planning, curriculum development, standardization efforts, technology transfer, and various editorial responsibilities, including a five-year term as Editor of *Computer Report*, a Farsi-language computing periodical. His technical publications include over 200 papers in journals and international conferences, a Farsi-language textbook, and an English/Farsi glossary of computing terms. His latest publications include two textbooks on computer arithmetic (Oxford, 2000) and parallel processing (Plenum, 1999). The textbook *Computer Architecture: From Microprocessors to Supercomputers* (Oxford, 2003) is forthcoming. Dr. Parhami is a Fellow of both the IEEE and the British Computer Society, a member of the Association for Computing Machinery, and a Distinguished Member of the Informatics Society of Iran for which he served as a founding member and President during 1979-84. He also served as Chairman of IEEE Iran Section (1977-86) and received the IEEE Centennial Medal in 1984.

Ding-Ming Kwai received the BS and MS degrees in Taiwan from the National Cheng Kung University, Tainan, and the National Chiao Tung University, Hsinchu, in 1987 and 1989, respectively, and the Ph.D. degree from the University of California, Santa Barbara, in 1997. He was with the Chung Cheng Institute of Technology, Taoyuan, Taiwan, as a reserve officer during 1989-91 and with the Hualon Microelectronics Corporation, Hsinchu, Taiwan, as a design engineer during 1991-93. He is now affiliated with the Worldwide Semiconductor Corporation, Hsinchu, Taiwan. His research interests include parallel processing, VLSI architectures, and fault-tolerant computing.