An Efficient Bit-Serial FIR Filter Architecture

Yong Ching Lim
Department of Electrical Engineering
National University of Singapore
Singapore 0511

Joseph B. Evans
Department of Electrical & Computer Engineering
University of Kansas
Lawrence, KS 66044-2228

Bede Liu
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

April 1993

Abstract

A new bit-serial architecture for implementation of high order FIR filters, as well as example FPGA and CMOS realizations are introduced. This structure exploits the simplicity of coefficients which consist of two power-of-two terms to yield efficient implementations. Quantization effects are discussed and a simple block scaling method for reducing rounding and truncation noise in high order filters is also presented.

This research is supported by the Office of Naval Research under Grant N00014-89-J1327, NSF Grant ECS87-13598, by an AT&T Bell Laboratories Graduate Fellowship, and by University of Kansas General Research allocation 3775-20-0038. Portions of this work were presented at ICASSP-90 in Albequerue, New Mexico.

1 Introduction

In recent years, considerable attention has been placed on the implementation of signal processing algorithms in VLSI [1, 2, 3, 4, 5, 7, 8, 9, 10, 13, 16, 17, 19, 21], ranging from full custom VLSI to general purpose digital signal processors. A variety of approaches to high speed implementation of FIR filters have been pursued.

Bit-serial processing techniques have been used in several applications [3, 14, 17, 18, 20]. We present a new bit-serial FIR filter building block which may be used to implement filters where each of the coefficient values is a sum or difference of several power-of-two terms, and which is particularly useful for the case where it is a sum or difference of only two power-of-two terms. Example designs based on field programmable gate arrays (FPGAs) as well as custom CMOS approaches will be used to illustrate the advantages of this architecture. The performance of the architecture on commercially available FPGAs, in particular, is comparable or superior to that of many custom filtering chips.

In binary arithmetic, multiplying a number by a power-of-two is just a matter of shifting, and consequently, the implementation may be simplified by using only a limited number of power-of-two terms. The minimum clock period may be as short as that required by a full adder and a latch. The improvements in speed and saving in silicon area are, however, achieved at the expense of a deterioration in the frequency response characteristics. The extent to which the frequency response deteriorates depends on the number of power-of-two terms used in approximating each coefficient value, the architecture of the filter, and the discrete space optimization technique used to derive the coefficient values. It has been demonstrated in [12] that -60dB of frequency response ripple magnitude can be achieved with two power-of-two terms for each coefficient value when the filter is realized in cascade form and the coefficient values are derived using mixed integer linear programming. These coefficient values are realized by appropriately positioning the summing points in the tapped delay line. A full custom design may be created by using transmission gates to control the summing points, while a semi-custom implementation may developed by metal layer interconnect positioning via contacts.

In addition to custom implementations, this architecture is well suited to field programmable gate arrays (FPGAs). Due to the prevalence of local routing in this architecture, both high performance and high density can be attained in a straightforward mapping to the FPGA building blocks and routing resources.

High filter length may be obtained by connecting the basic building blocks in series. Due to the large number of rounding and truncation operations in high order filters, quantization noise effects may be serious. A new block scaling method is introduced to reduce these noise effects.

2 Architecture

In a bit-serial implementation, each delay element of the filter is replaced by an M-stage single-bit shift register, as shown in Figure 1, where M is the wordlength of the filter. If the coefficient value is an integer power-of-two, the multiplier can be replaced by a barrel shifter. There is a more efficient method, however, for implementing a coefficient value which is an integer power-of-two. It will be shown later that moving the adder k places to the right achieves the same effect as would be achieved by a coefficient value of 2^{-k}.

2.1 General Structure

Consider the bit-serial summation of S_n and U_n to produce V_n, as shown in Figure 2, where S_n, U_n, and V_n are given by

$$S_n = \sum_{m=0}^{M-1} s_n(m)2^m, \quad s_n(m) = 0, 1 \tag{1}$$

$$U_n = \sum_{m=0}^{M-1} u_n(m)2^m, \quad u_n(m) = 0, 1 \tag{2}$$

$$V_n = \sum_{m=0}^{M-1} v_n(m)2^m, \quad v_n(m) = 0, 1 \tag{3}$$

The variables $s_n(m)$, $u_n(m)$, and $v_n(m)$ do not exist for $m < 0$ and $m > M - 1$. In Figure 2, $u_n(m)$ is clocked into a shift register, least significant bit first, and $s_n(m)$ is added to $u(m - k)$ at the k^{th} stage producing $v(m - k)$. The adder is a single-bit full adder. Hence,

$$v_n(m) = u_n(m) + s_n(m + k) \tag{4}$$

The requirements for the existence of $u_n(m)$, $v_n(m)$, and $s_n(m + k)$, and the possibility of a carry being generated are ignored at this moment; these will be discussed in the next subsection. Multiplying both sides of (4) by 2^m and summing from $m = 0$ to $M - 1$, we have

$$V_n = U_n + 2^{-k} \sum_{m=k}^{M-1+k} s_n(m)2^m \tag{5}$$

Hence,

$$V_n = U_n + 2^{-k} S_n - 2^{-k} \sum_{m=0}^{k-1} s_n(m)2^m \tag{6}$$

and

$$V_n = U_n + 2^{-k} S_n, \quad \text{if } s_n(m) = 0 \text{ for } m = 0, \ldots, k - 1 \tag{7}$$

Suppose that S_n is derived from an R-bit data word D_n, where

$$D_n = \sum_{m=M-R}^{M-1} r(m)2^m, \quad r(m) = 0, 1 \text{ and } R \leq M \tag{8}$$

and $S_n = D_n$, then it is necessary that $s_n(m) = 0$ for $m = 0, \ldots, M-R-1$. Thus, if $M - R \geq k$, then $V_n = U_n + 2^{-k} S_n$. We shall refer to R as the input signal wordlength. Note that k is the equivalent "coefficient wordlength". Hence, if the signal wordlength plus the "coefficient wordlength" is less than or equal to the internal wordlength of the filter, the coefficient multiplier 2^{-k} can be implemented using (4).

2.2 Control Unit and Adder Design

In bit-serial architectures, the word-boundary must be identified. The summation operation $s_n(m) + u_n(m-k)$ should not be performed if $m - k < 0$ since $u_n(m+k)$ for $m - k < 0$ is undefined. When $m - k < 0$, $u_n(m+k)$ is in fact part of U_{n-1}; S_{n-1} should be sign extended (assuming two's complement representations are being used) and added to $u_n(m+k)$ when $m - k < 0$. This timing can easily be controlled by a set of timing signals. Let the j^{th} timing signal at time i be $\theta_j(i)$. The logic level of $\theta_0(i)$ is time invariant and is always equal to 1. Except during $i = 0$, the logic levels of $\theta_j(i)$ for $j = 2, \ldots, M-1$, are given by

$$\theta_j(i) = \theta_{j-1}(i-1) \tag{9}$$

At the word-boundary, i.e. when $i = 0$, $\theta_j(i)$ for $j = 2, \ldots, M-1$, are reset to zero. Whether $s_n(m)$ should be added to $u_n(m+k)$ or whether the most significant bit of S_n should be sign extended and then added to $u_n(m+k)$ is controlled by $\theta_j(i)$. Those adders associated with the coefficients whose values are 2^{-j} should add $s_n(m)$ to $u_n(m+k)$ when $\theta_j(i) = 1$; S_{n-1} should be sign extended and then added to $u_n(m+k)$ when $\theta_j(i) = 0$.

Since the weight of $v_n(m+1)$ is twice that of $v_n(m)$, the carry generated by the right-hand side of (4) should be added to $v_n(m+1)$. Hence, the complete equation for (4) is

$$v_n(m) + c_n(m) = u_n(m) + s_n(m+k) + c_n(m-1), \tag{10}$$

where $c_n(m)$ is the carry generated from the summation operation $u_n(m) + s_n(m+k) + c_n(m-1)$. The adder structure is shown in Figure 3. One of the inputs of the adder is $u(m-k)$ and the other input is either $s(m)$ or $\bar{s}(m)$ depending on whether the coefficient is positive or negative. The multiplexer and the latch perform the sign extension under the control of b_1, which is connected to the appropriate signal line $\theta_j(i)$. Since the carry bits should not be allowed to propagate past the word-boundaries, the control line b_2 is used to clear the delay at word-boundaries.

3 Quantization Noise Effects

3.1 Architecture Performance

If $R+k$ is larger than the wordlength of the filter, the signal wordlength must be truncated, producing roundoff noise. When a digital filter is implemented using nontrivial multipliers, the roundoff noise power is $Q^2/12$ [15, 18] for each coefficient, where Q is the quantization step size. This noise power is derived based upon the assumption that the probability density function of the rounding error is rectangularly distributed. When a filter is implemented using the structure proposed in this paper, however, the roundoff noise properties are quite different. Those coefficients with $R+k \leq M$ do not introduce roundoff noise. Those coefficients with $R+k >> M$ generate roundoff noise with the same noise power as that generated by a nontrivial multiplier. When $R+k-M$ is small, the roundoff noise consists of a DC bias and a broadband AC component. The DC bias and the AC noise power for small values of $R+k-M$ are tabulated in Table 1. It can be seen from Table 1 that the AC noise power is less than $Q^2/12$ and approaches $Q^2/12$ for $R+k >> M$.

The large number of rounding or truncation processes in the implementation of high order FIR filters can render the filter output noisy. A commonly used method for reducing the roundoff noise of high order FIR filter is to implement the filter using a cascade structure. In a cascade realization, the roundoff noise introduced in the preceding stages is partially removed by the stopband response of the succeeding stages. Although the cascade realization reduces the total roundoff noise power, the reduction in roundoff noise is achieved mainly in the reduction of the noise power density in the stopband of the filter. The signal to noise ratio in the passband cannot be improved by a cascade realization since all the cascaded sections must pass the signal and noise in the passband. Hence, a cascade realization is not effective in reducing the roundoff noise in wideband filters. A simple block scaling method for reducing the roundoff noise over the entire frequency spectrum for filters with arbitrary bandwidth addresses this problem.

3.2 A Block Scaling Method

Figure 4 is a typical impulse response sequence of an FIR filter. The only important observation to be noted is that only a few samples of the impulse response sequence at the middle have large magnitudes; the magnitudes of the remaining samples are small. Let $H(z)$ be the z-transform transfer function of the filter (assuming symmetry),

$$H(z) = h(0) + \sum_{n=1}^{(N-1)/2} h(n)(z^n + z^{-n}). \tag{11}$$

Assume that rounding is performed after every multiplication and that the noise power of each rounding process is σ^2. The total noise power at the output is $(1+2L+2M+2K)\sigma^2$. The impulse response sequence, $h(n)$, can be sectioned into three (or more) blocks as shown in Figure 4, based upon their magnitudes. Suppose there are $2L+1$ large samples, $2M$ small samples, and $2K$ very small samples. The small and very small samples may be made large by multiplying them by

suitable scale factors. Suppose we define

$$h(n) = s_1 h_1(n) \quad \text{for } n = L+1, ..., L+M \tag{12}$$

$$h(n) = s_2 h_2(n) \quad \text{for } n = L+M+1, ..., L+M+K \tag{13}$$

where s_1 and s_2 are appropriate scale factors. $H(z)$ can then be rewritten as

$$\begin{aligned} H(z) &= h(0) + \sum_{n=1}^{L} h(n)(z^n + z^{-n}) + s_1 \sum_{n=L+1}^{L+M} h_1(n)(z^n + z^{-n}) \\ &+ s_2 \sum_{n=L+M+1}^{L+M+K} h_2(n)(z^n + z^{-n}) \end{aligned} \tag{14}$$

With these scale factors, $H(z)$ may be implemented using the structure shown in Figure 5. The symmetry of the impulse response sequence may be used to reduce hardware complexity but it is not used in the structure shown in Figure 5 purely for expository convenience; the structure can easily be modified to make use of the symmetry.

The total noise power at the output of the block scaled implementation is $(2L+1)\sigma^2 + 2(Ms_1^2 + 1)\sigma^2 + 2(Ks_2^2 + 1)\sigma^2$. The noise power decreases asymptotically to $(2L+5)\sigma^2$ as $Ms_1^2 + Ks_2^2$ decreases. Hence, it is advantageous to choose s_1 and s_2 as small as possible. As s_1 and s_2 decrease, $h_1(n)$ and $h_2(n)$ increase. In order to avoid overflow,

$$\sum_{n=L+1}^{L+M} |h_1(n)| \leq 1 \qquad \text{and} \qquad \sum_{n=L+M+1}^{L+M+K} |h_2(n)| \leq 1. \tag{15}$$

Hence, s_1 and s_2 should be chosen so that

$$s_1 \geq \sum_{n=L+1}^{L+M} |h(n)| \quad \text{and} \quad s_2 \geq \sum_{n=L+M+1}^{L+M+K} |h(n)|. \tag{16}$$

For high order filters, the magnitude of $h(n)$ for large n is several orders of magnitude less than that for small n. Hence, the scale factors s_1, s_2,..., etc. are small. The noise reduction effect is significant. Furthermore, this method reduces the noise spectrum uniformly across the entire spectrum, so that it is equally effective for wideband as well as narrowband filters.

4 Implementation

The proposed bit-serial processing technique is eminently suitable for implementing FIR filters whose coefficient values are sums or differences of power-of-two terms, particularly for the case of two power-of-two terms. Figure 6(a) shows the floor plan of a cell consisting of two full adders, implemented as previously indicated, and an M-stage single-bit shift register; *s* is the input data signal, *u* is the input from the previous tap, and *v* is the output to the next tap. This module implements a single FIR filter tap, as shown by the equivalent circuit model in Figure 6(b).

4.1 FPGA Implementation

The efficiency of this architecture is demonstrated by implementation on commercially available field programmable gate arrays (FPGAs), in particular the Xilinx XC3100 series components. This approach is illustrated in Figure 7, where the interconnection pattern for a typical filter tap is shown. Each block corresponds to a single configurable logic block (CLB), and most data signals are routed locally. Control signals for implementing sign extension are distributed using the horizontal long lines of the array (denoted by the dotted lines) and are applied to the input signal in the centrally located control block. Automatic generation of the FPGA configuration information starting from the specification of the filter response is straightforward, as only the interconnection pattern of the adder and delay element inputs change with filter characteristics.

Using 10 bit coefficient words, a filter tap has been implemented using 7 CLBs. A filter with 60 taps could thus be implemented on a single XC3195 FPGA, with sampling rates of approximately 5 MHz. If present trends in integrated circuit technology continue, filters implemented in FPGAs with 600 taps and sampling rates of 20 MHz should be attainable by the end of the decade using this structure.

4.2 Custom CMOS Implementation

This bit-serial architecture can be used for implementing both fully programmable filters, or mask programmable semi-custom designs. In the fully electrically programmable structure, the summing points are implemented using transmission/pass gates which are controlled by coefficient registers unique to a given tap. In a mask programmable implementation, the summing points are selected using the metal layer mask. The mask programmable design can provide superior area/speed performance due to the reduced circuit complexity, but only at the expense of reduced flexibility.

CMOS implementations have been used to emphasis the advantages of this architecture. These implementations were designed in a 2.5 micron, single level metal, single level polysilicon CMOS process. Some details of the implementations are given in Table 2. The architecture is such that these configurations can be used to realize arbitrary filter responses.

The fully programmable chip is arranged in two independent sections, an east and a west side, each with 25 taps stacked vertically. Buffers are inserted in the center of each stack, so that the IC appears to be divided into quadrants. The two sides have independent clocking and data inputs and outputs; the pin connections may be used to cascade the sections to obtain a 48 tap filter with a single integrated circuit.

The mask programmable chip is arranged in three sections, each with 48 taps stacked vertically. Buffers are used in the center of each stack to reduce fan-out and control signal skew problems. The sections share the master clock but have independent data inputs and outputs. The pin connections may be used to cascade the sections to obtain a 144 tap filter with a single integrated circuit.

Clock distribution is accomplished via buffers at the corners of each quadrant; these buffers in turn drive the buffers at both ends of each tap row. This scheme is designed to minimized clock skew problems and limit the buffer loading. In the case of the fully programmable chip, independent clocks are used for normal filtering operation and coefficient loading, so as to further minimize clock line loading.

The order of the filter can be expanded by using either direct form expansion or cascade form expansion. In the case of the direct form expansion, the data input is broadcast to all filter sections, and the output of a given section is tied to the sum input of the next filter section. In the case of cascade expansion, the output of a given filter section is simply tied to the input of the next section.

These CMOS implementations were developed for a relatively primitive fabrication process; considerably higher densities should be attainable in a more modern process. These implementations will particularly benefit from additional metal layers, as clock and signal routing was severely constrained. Further, implementation on a modern custom gate array architecture should be straightforward, and provide extremely high density and performance.

The current implementations compare favorably to those in [5, 6, 11, 19, 21, 22] when normalized for the differences in fabrication technology. This is illustrated in Table 3, where the "score" is calculated according to the sampling rate multiplied by the number of taps per unit area, with normalization for the particular technology used. Note that this simplistic comparison does not consider differences in word length or coefficient codings, but it does provide some insight into the results of the various design approaches.

5 Conclusions

A new bit-serial FIR digital filter structure which is eminently suitable for efficient VLSI implementation of filters whose coefficient values are sums or differences of power-of-two terms has been presented. By exploiting the restriction on the coefficient values, the new architecture yields extremely efficient FPGA and custom implementations. Example FPGA and CMOS implementations employing this architecture were discussed. Further, a block scaling approach for reducing the roundoff noise in high order FIR filters such as might be implemented using these architectures was presented. In contrast with cascade form realization which reduces the roundoff noise in the stopband only, this block scaling approach reduces the roundoff noise over the entire frequency spectrum.

References

[1] M. A. Bayoumi, G. A. Jullien, and W. C. Miller. A VLSI implementation of residue adders. *IEEE Trans. Circuits and Syst.*, CAS-34:284–288, March 1987.

[2] P. R. Cappello, editor. *VLSI Signal Processing*. IEEE Press, 1984.

[3] P. Denyer and D. Renshaw. *VLSI Signal Processing: A Bit-Serial Approach*. Addison-Wesley Publishing Co., 1985.

[4] J. Ginderdeuren, H. DeMan, A. Delaruelle, and H. Wyngaert. A digital audio filter using semiautomated design. In *ISSCC Dig. Tech. Pap.*, pages 88–89, 1986.

[5] R. Hartley, P. Corbett, P. Jacob, and S. Karr. A high speed FIR filter designed by compiler. In *IEEE Cust. IC Conf.*, pages 20.2.1–20.2.4, May 1989.

[6] M. Hatamian and S. Rao. A 100 MHz 40-tap programmable FIR filter chip. In *IEEE Int. Symp. Circuits and Syst.*, pages 3053–3056, May 1990.

[7] R. Jain, P. Yang, and T. Yoshino. Firgen: A computer-aided design system for high performance FIR filter integrated circuits. *IEEE Trans. Signal Processing*, 39(7):1655–1668, Jul 1991.

[8] S. Y. Kung, R. E. Owen, and J. G. Nash, editors. *VLSI Signal Processing II*. IEEE Press, 1986.

[9] S. Y. Kung, H. J. Whitehouse, and T. Kailath, editors. *VLSI and Modern Signal Processing*. Prentice-Hall, Inc., 1985.

[10] H. K. Kwan. A multi-output first-order digital filter structure for VLSI implementation. *IEEE Trans. Circuits and Syst.*, CAS-32:973–974, Sept 1985.

[11] J. Laskowski and H. Samueli. A 150-Mhz 43-tap half-band FIR digital filter in 1.2-μm CMOS generated by compiler. In *IEEE Cust. IC Conf.*, pages 11.4.1–11.4.4, May 1992.

[12] Y. C. Lim and B. Liu. Design of cascade form FIR filters with discrete valued coefficients. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-36:1735–1739, Nov 1988.

[13] H. Lin. New VLSI systolic array design for real-time digital signal processing. *IEEE Trans. Circuits and Syst.*, CAS-33:673–676, June 1986.

[14] J. G. McWhirter et al. Multibit convolution using a bit level systolic array. *IEEE Trans. Circuits and Syst.*, CAS-32:95–99, Jan 1985.

[15] A. Oppenheim and R. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[16] R. M. Owens. Techniques to reduce the inherent limitations of fully digit on-line arithmetic. *IEEE Trans. Comput.*, C-32(4):406–410, Apr 1981.

[17] N. R. Powell and J. M. Irwin. Signal processing with bit-serial word-parallel architectures. *SPIE Real-Time Signal Processing*, 154:98–104, 1978.

[18] L. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[19] P. Ruetz. The architectures and design of a 20-MHz real-time DSP chip set. *IEEE J. Solid State Circuits*, 24(2):338–348, Apr 1989.

[20] H. J. Sips. Bit-sequential arithmetic for parallel processors. *IEEE Trans. Comput.*, C-33(1):7–20, Jan 1984.

[21] F. Yassa, , J. Jasica, et al. A silicon compiler for digital signal processing: Methodology, implementation, and applications. *Proc. IEEE*, 75(9):1272–1282, Sep 1987.

[22] T. Yoshino, , R. Jain, et al. A 100-MHz 64-tap FIR digital filter in 0.8 μm BiCMOS gate array. *IEEE J. Solid State Circuits*, 25(6):1494–1501, Dec 1990.

Table 1: The DC Bias and AC Noise Power for $P + k - M = 1, 2, 3$

$P + k - M$	DC Bias	AC Noise Power
1	$0.25Q$	$0.0625Q^2$
2	$0.125Q$	$0.0781Q^2$
3	$0.0625Q$	$0.0820Q^2$

Table 2: FIR Filter ASIC's (2.5 μm CMOS)

	Fully Programmable	Mask Programmable
input data	9 bits	9 bits
coefficients	9 bits + 2 sign bits	9 bits + 2 sign bits
filter length	48 taps	144 taps
maximum sampling rate	5 MHz	6 MHz
maximum clock rate	50 MHz	60 MHz
transistors	29396	30108
active area	5.838 mm by 5.840 mm	5.508 mm by 5.585 mm
die size	7 mm by 7 mm	7 mm by 7 mm

Table 3: FIR Filter ASIC Comparison

	Taps	Area (mm^2)	Rate (MHz)	Technology (μm^2)	Score
Mask Programmable	144	30.762	6.0	2.5	438.9
Fully Programmable	48	34.094	5.0	2.5	109.9
Laskowski [11]	43	40.95	150.0	1.2	139.3
Yoshino [22]	64	48.65	100.0	0.8	33.68
Ruetz [19]	64	225	22.0	1.5	21.12
Yassa [21]	16	17.65	30.0 (estimate)	1.25	53.12
Hartley [5]	4	25.8 (estimate)	37.0	1.25	11.20
Hatamian [6]	40	22	100.0	0.9	132.5

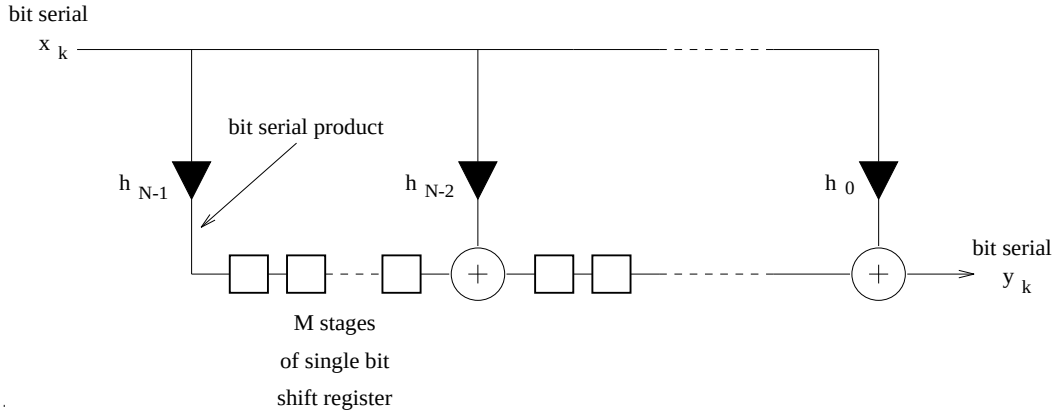

Figure 1: A Bit-Serial Inverted FIR Filter

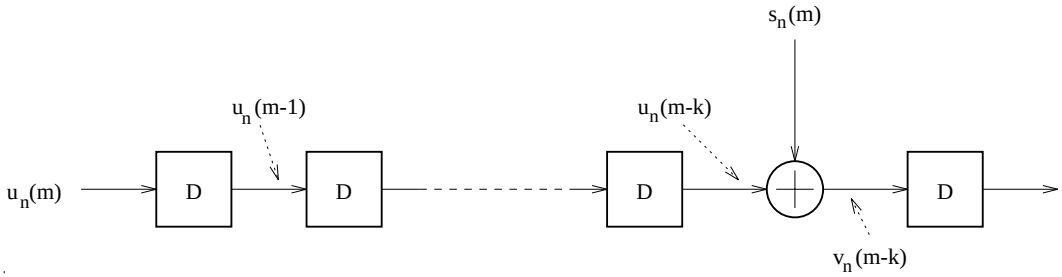

Figure 2: Bit Serial Summation, $v_n(m-k) = s_n(m) + u_n(m-k)$

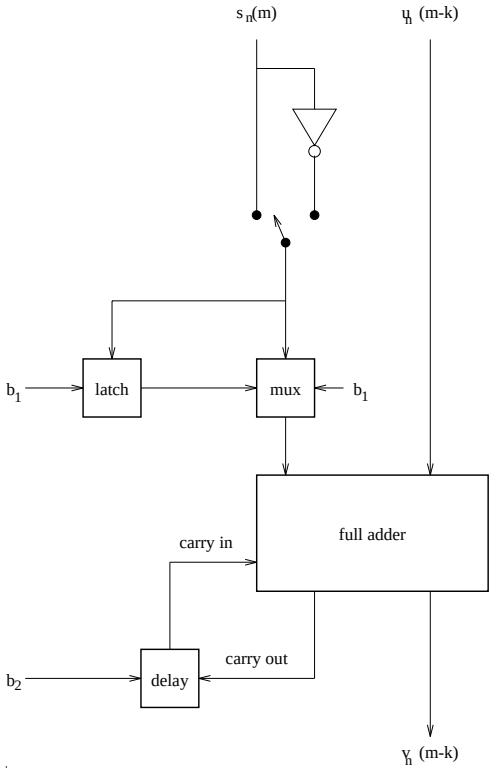

Figure 3: FIR Filter Tap Adder Unit

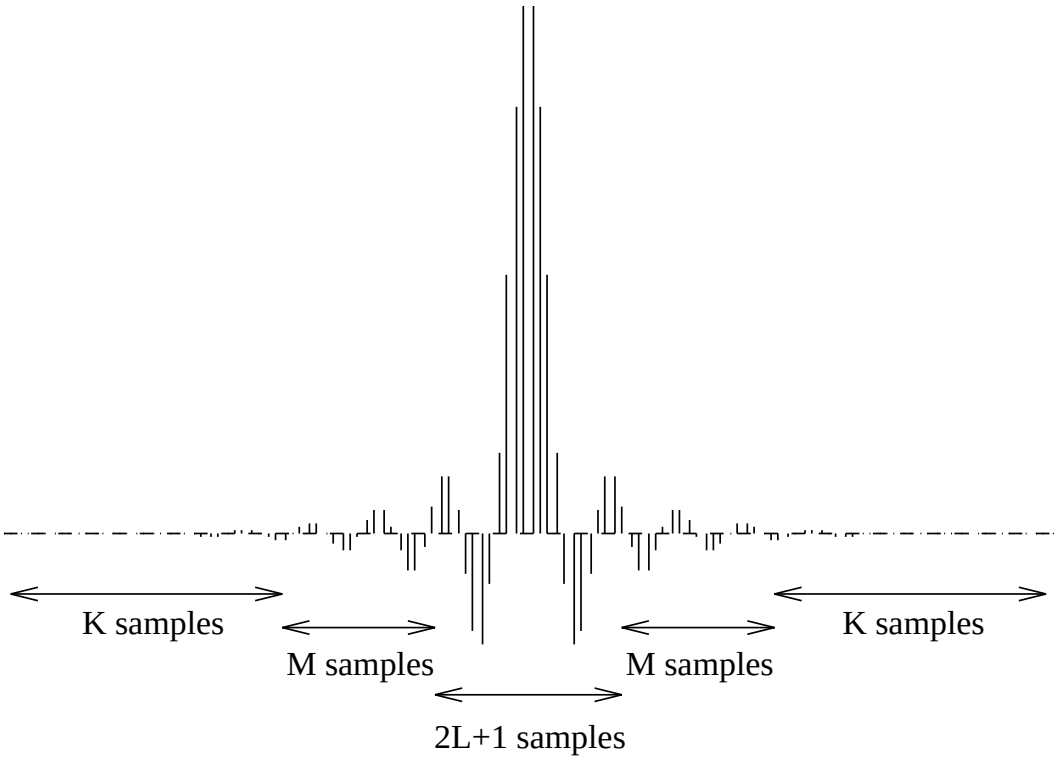

Figure 4: A Typical FIR Filter Impulse Response

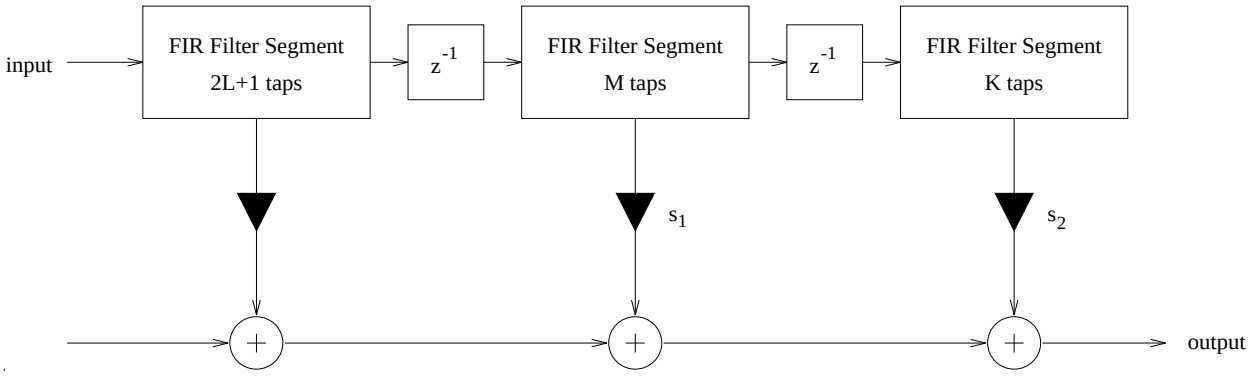

Figure 5: The Block Scaling Structure

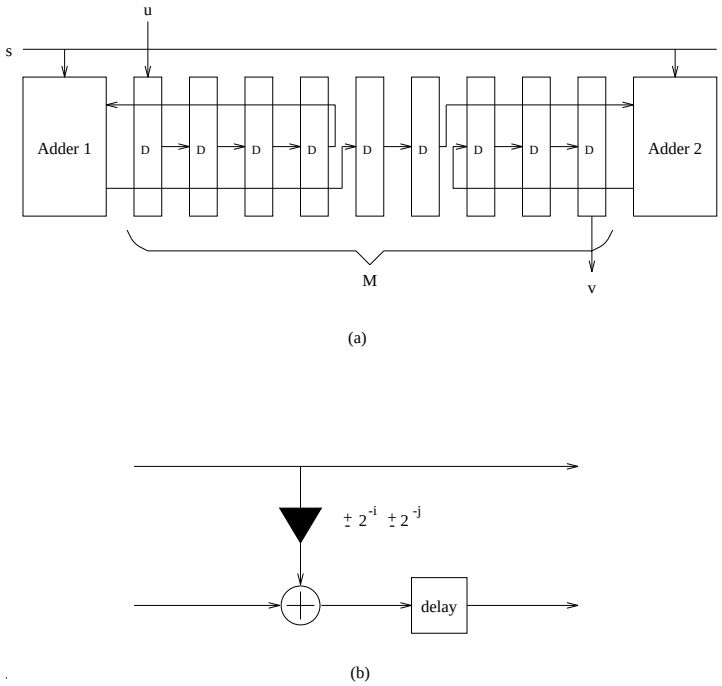

Figure 6: Filter Tap Circuit: (a) Floor Plan (b) Functional Model

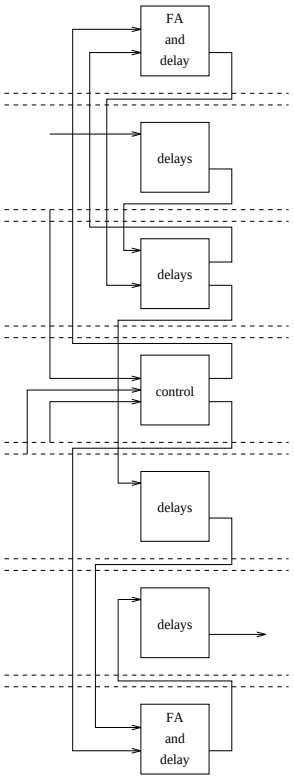

Figure 7: FPGA Filter Tap Implementation