A New Area-efficient FIR Filter Design Algorithm by Dynamic Programming

Juan Zhao
Pillar of Information Systems Technology and Design
Singapore University of Technology and Design
Singapore
juan_zhao@mymail.sutd.edu.sg

Yujia Wang, Jiajia Chen* and Feng Feng*
Pillar of Information Systems Technology and Design
*Pillar of Engineer Product Development
Singapore University of Technology and Design
Singapore
jiajia_chen@sutd.edu.sg

***Abstract*— Finite impulse response (FIR) digital filter is a commonly adopted signal processing unit in digital signal processing, due to its stability and easy implementation for linear phase response. To reduce the area complexity and power consumption of the filters, researchers have proposed algorithms to optimize the expression of coefficients with the reduced number of non-zero digits or power-of-two terms. This paper presents a new optimization algorithm based on elegant dynamic programming approach to minimize the number of non-zero digits in coefficient set, which yields to low logic operator cost in the filter circuit implementation. The proposed algorithm utilized the knapsack method from dynamic programming to effectively remove the redundant nonzero digits in the coefficients. Experiment results on benchmark filters show that the proposed algorithm can synthesize FIR filter coefficient with the reduced area complexity. Compared with two competing methods, the proposed algorithm can design FIR filters with an average of 14.49% to 47.73% reduced full adder cost over the competing methods.**

Keywords—finite impulse response (FIR) filter; dynamic programming; digital signal processing.

I. INTRODUCTION

FINITE impulse response (FIR) digital filter is a pivotal digital component in digital signal processing (DSP), communications, controls and automations. Driven by more stringent size and power budget in mobile applications, FIR filter designs require design techniques for combinatorial optimization of formidable complexity with drastically different emphasis from software compilers to general purpose DSPs [1] [2]. Therefore, it is an emergent topic to develop new optimization methods that will facilitate synthesis of efficient hardware solutions. The computer-aided design methodologies to be studied in this research form the basis for exploitation by high-level synthesis tools for the design of high-speed, low complexity and reconfigurable application-specific digital filters [3].

A discrete-time FIR filter of order N can be expressed as follows:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots + b_N x[n-N] = \sum_{i=0}^{N} b_i \cdot x[n-i] \quad (1)$$

where $x[n]$ and $y[n]$ are the inputs and outputs respectively. b_i is the impulse response of the filter which are also named as coefficients.

Compared with infinite impulse response (IIR) digital filter, FIR filter requires no feedback and is inherently stable. However, the filter order of FIR filter is larger and hence more multiplications are required resulting in higher area complexity and power consumption. Thus, to reduce the cost, researchers have proposed algorithms [4]-[8] to design hardware efficient FIR filters, such as the multiplierless multiple constant multiplication (MCM) [8].

In multiplierless based FIR filter design, each M-bit coefficient b_i is expressed as the sum of power-of-twos, i.e.

$$b_i = \sum_{j=0}^{M-1} k \cdot 2^j \quad (2)$$

where k equals to either 0 or 1. Therefore, area intensive multiplications can be simplified to shifts and additions, which only require shifters and adders. For example, if the two coefficients are 11001 and 101001, its multiplierless design to produce these two coefficients is shown in Fig. 1. As the hardwired shifters encounter no cost, the area complexity of the filter is dominated by the adders. The number of adders required depends on the number of non-zero digits existing in all the coefficients. As a result, Canonic Signed Digit (CSD) representation is often adopted to represent the coefficients because CSD can represent a value using the least number of non-zero digits. The next problem is how to synthesize the FIR filter coefficient values with the least number of non-zero digits in CSD form, while the required frequency domain specifications are met. Recent approaches [4] [5] have been proposed to reduce the non-zero digits in the coefficients set. These methods are generally proposed by searching and trying with different coefficients until the best solution is found. Because of the lack of elegant optimization approach, optimality of the results by these methods can hardly be guaranteed.

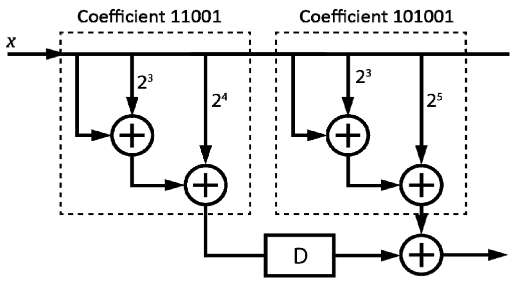

Fig.1. Implementation of multiplierless multiplication between input sample and filter coefficients.

In this paper, therefore, a new algorithm is proposed to minimize the number of non-zero digits of FIR coefficients using dynamic programming optimization. The dynamic programming problem is solved using Memoization, which will be illustrated in the following sections with details.

II. Dynamic Programming to Reduce Non-zero Digits

Dynamic programming is a method of solving complex problem by dividing it into sub-problems, computing each sub-problem once and storing their values to be reused [8]. This technique of storing solutions to sub-problems instead of re-computing them is called "memoization". Its memory-based data structures avoid repetition, thus optimize the computing complexity. In our proposed algorithm, we choose the required passband ripple δ_p and the stopband attenuation δ_s as the optimization constraints. Therefore, the probleming of design area efficient FIR filter is transformed into synthesizing finite-precision filter coefficients with the minimized number of non-zero digits while the actually produced passband ripple $\delta_p' \le \delta_p$ and stopband attenuation $\delta_s' \ge \delta_s$. This optimization using dynamic programming includes iterative loops. Let *T* be the updated minimum number of non-zero digits in the coefficients, each loop can produce a valid solution when the newly found number of non-zero digits *t* is less than *T* with the design specifications are fulfilled.

III. Infinite-precision Coefficient Generation and The Proposed Algorithm

A. Infinite-precision coefficients generation and truncation

Taken the filter design specifications as inputs, Parks-McClellan (PM) algorithm [9] generates infinite-precision coefficients set in decimal form with the least number of filter order *N*. The infinite long coefficients will first be converted to CSD form, and truncated to *L*-bit. In our method, *L* is chosen to be 24 as the 24-bit truncated coefficients from the infinite precision can always produce the designed filter fulfilling all the specifications in all our experiments. Once the *N*+1 24-bit coefficient set *C* are generated, the proposed algorithm synthesizes the FIR filter using the modified Knapsack method from dynamic programming.

B. Modified knapsack method in FIR filter coefficient synthesis

The original Knapsack problem [8] is defined as:

Given a set of items and each with a weight and a value, determine the collection of a number items so that the total weight is less than or equal to a given constraint and the total value is as large as possible.

In the proposed algorithm, every non-zero digit in the coefficient set *C* after truncation is initially treated as one item, i.e. i_1, i_2, … i_T. The weight of one item is defined as the maximum impact to the frequency response when removing this item. The proposed algorithm also defines δ_p and δ_s as constraints, and the goal is to find the optimal collection of *v* non-zero digits that could be removed from the coefficient set *C* while the requirements in δ_p and δ_s are fulfilled. To minimize the non-zero digits remaining in the coefficient set, i.e. *T*−*v*, *v* is the total value in our algorithm which will be maximized.

The proposed algorithm based on dynamic programming follows the two steps below.

Step 1: Construct the Memoization table

A Memoization table to store the coefficients through non-zero digits elimination is constructed as shown in Table 1. The first column is filled with the *T* items. The first row is filled with $\delta_p(\alpha)$ and $\delta_s(\alpha)$, which are *α*% more stringent than δ_p and δ_s following the principles of dynamic programming. Taking δ_p=0.01dB and δ_s=60dB as an example, the 10% more stringent $\delta_p(10)$=0.009dB and $\delta_s(10)$=66dB. The more stringent rate *α*% falls in a range of percentages. Based on our experiments, this percentage range is selected to be between 0% to 10% with a step size at 1% in this paper, so that every more stringent requirement can have some items removable during the process.

Table 1. Memoization table

	10%	9%	8%	7%	6%	5%	4%	3%	2%	1%	0%
i_1										→	
i_2	←									→	
i_3					⋯						
⋯					⋯						
i_T										→	

Step 2: Non-zero digit elimination based on dynamic programming

In the *T* by 11 Memoization table, each entry *C*(*l*, *m*) at the *l*th row and *m*th column is the coefficient set with the *l*th item removed from C if $\delta_p(m)$ and $\delta_s(m)$ can be fulfilled. If not, that entry is left as blank. Apparently, if one item is removable at the kth column, it is surely removable at the columns from k+1 until column 11, so the filling can directly jump to the next row. Every time when an item is removed, the coefficient set is updated with that item eliminated and this new set is used for filling check in the next entry. This filling process starts from row 1 column 1 with the original coefficient set C and move on from left to right, top to bottom as arrows demonstrate in Fig.1.

When we reach the last cell in the right bottom corner eventually, the value v of that cell is exactly the maximal number of removable non-zero digits from C. With this, we thus obtain the optimized coefficients set with the least number of non-zero digits remaining in C.

C. Pseudo code of the proposed algorithm

The following pseudo code summarizes the flow of the proposed algorithm.

```
FIR(ori_coeff, L, $\delta_p$, $\delta_s$, $\alpha$) {
 begin
   ori_csdMatrix=DecToCSDMatrix (ori_coeff, L);
   num_none_zero_digit=countDigit(ori_csdMatrix);
   position_none_zero_digit=recordDigit(ori_csdMatrix);
   weights=toArray( $\delta_p$ ( $\alpha$ ), $\alpha$ );
   DP=zeros(num_none_zero,weights);
   DP_csdMatrix=zeros(num_none_zero,weights, size(ori_csMatrix));
   //initialize a table to store updated coefficient set
   forall i $\in$ (1,num_none_zero_digit )
     forall j $\in$ (0, weights.length)
       k = j;
       forall k>=0
         matrix = DP_csdMatrix(i-1, k);
         matrix=changeDigitToZero(position_none_zero_digit(i));
         //update coefficient set and set the number in i^th position zero
         if (freq_resp_satisfied(matrix, weights(j)))
           DP(i, j) = DP(i, k)+1; DP_matrix = matrix;
         end
       end
       if(freq_resp_satisfied(DP_csdMatrix(i-1,j),weights(j)))
         //DP_csdMatrix(i-1,j)satisfies passband ripple
         DP(i, j) = max( DP(i-1, j) , DP(i, j));
         DP_csdMatrix(i, j) = update(DP(i, j));
       end
       if(freq_resp_satisfied(ori_csdMatrix, weights(j)))
         //ori_csdMatrix satisfies passband ripple
         DP(i, j) = 0; DP_csdMatrix(i, j) = ori_csdMatrix;
       else
         DP(i, j) = -1; DP_csdMatrix(i, j) = ori_csdMatrix;
       end
     end
   end
   csdMatrix = DP_csdMatrix(num_none_zero, weights);
   coeff=CSDMatrixToDec(csdMatrix);
   return coeff;
 end}
```

Fig.2. Pseudo code for the proposed algorithm

The function **FIR()** takes the inputs of decimal coefficient set *ori_coeff*, coefficient length L for truncation, passband ripple δ_p , stopband attenuation δ_p and incremental step size α. It produces the final coefficient set with the minimized number of non-zero digits in decimal form. The function **DecToCSDMatrix()** takes coefficient set in decimal form and transforms it into CSD form. The function **countDigit()** and **recordDigit()** both take coefficient set in CSD form as input, **countDigit()** calculates the total number of non-zero digits of input and **recordDigit()** indexes non-zero digits into an array called *position_none_zero_digit*. The function **toArray()** takes δ_p and α as inputs and returns an array of passband ripples whose step size is α. The function **changeDigitToZero ()** removes one non-zero digit and return an updated coefficient set. The function **freq_resp_satisfied**() takes inputs of a coefficient set and passband ripple and returns true if the coefficient set satisfies the passband ripple.

IV. Experimental Results and Discussion

Five FIR filter benchmarks are employed in this section to evaluate the effectiveness of the proposed algorithm to design low complexity FIR filter. The required specifications δ_p, δ_s in decibel form (dB) and filter type of these benchmark filters are listed in Table 2. Low-pass filter is denoted as *lp,* and high-pass filter is denoted as *hp.*

Table 2. Benchmark filter specifications

	FIR1[6]	FIR2[7]	FIR3[11]	FIR4[11]	FIR5[6]
δ_p (dB)	0.105	0.004	0.0087	0.0278	0.051
δ_s (dB)	60	70	60	50	20
type	*lp*	*lp*	*lp*	*lp*	*hp*

The algorithms in comparison with the proposed method are Mixed Integer Linear Programming (MILP) [5] which is one commonly cited methods in this topic and Multiple Constant Multipliers/Accumulators with faithfully rounded Truncation (MCMAT) [4] which is a recently published method. MILP optimizes the filter coefficients directly in subexpression space for a given specification. MCMAT jointly considers the optimization of coefficient length and hardware resources without sacrificing the frequency response. In the comparison, the designed filters by the propose algorithm and two competing methods have been verified to make sure all the desired frequency responses are met. For example, the frequency response of the designed FIR1 by the proposed algorithm is shown in Fig. 3, which met the requirements as stated in Table 2.

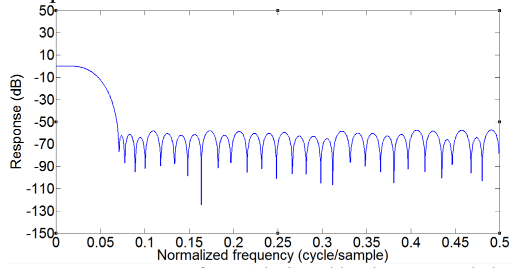

Fig.3. Frequency response of FIR1 designed by the proposed algorithm

The number of non-zero digits T in the synthesized coefficient set and full adder cost are two parameters we used to evaluate the area complexities of the designs by these methods. The filter order N and number of non-zero digits in coefficient set are presented in Table 3.

Table 3. Number of non-zero digits (T) in the synthesized coefficient sets and filter order (N) by the proposed algorithm, MILP,and MCMAT.

	Filter	FIR1[6]	FIR2[7]	FIR3[11]	FIR4[11]	FIR5[6]
Proposed	T	101	73	48	39	27
	N	60	43	38	28	31
MILP	T	411	302	256	86	84
	N	60	43	38	28	31
MCMAT	T	109	72	58	41	29
	N	60	43	38	28	31

From Table 3, both the proposed algorithm and MCMAT significantly reduced the number of non-zero digits (T) compared with MILP, whereby the proposed algorithm effectively reduced T by 72.54% over MILP on average. Compared between the proposed algorithm and MCMAT, the proposed algorithm designed the filters with an average of 9.09% less non-zero digits over the designs by MCMAT.

To compare the area complexity of the designed filters with a finer cost metric, full adder cost to implement the designed filters are computed and presented in Table 4. Ripple carry adder (RCA) is assumed in this cost estimate for its linear relation of area complexity with the length of the adder. The complexity of one half adder is assumed to be the half of the complexity of a full adder, so the fractional full adder cost number in Table 4 is because of the odd number of half adders used in the particular filter circuit.

Table 4. Full adder cost of the filters designed by the proposed algorithm, MILP and MCMAT.

Filter	FIR1[6]	FIR2[7]	FIR3[11]	FIR4[11]	FIR5[6]
Proposed	192.5	153.5	128.5	85	70.5
MILP	214.5	246.5	287.5	242	239
MCMAT	213.5	179	157	85.5	78

From Table 4, the proposed algorithm reduced an average of 47.73% full adder cost compared with MILP. Compared with MCMAT, the proposed algorithm also designed the filters with an average of 14.49% less full adder cost. The reduced non-zero digits and hence the less full adder cost by the proposed algorithm is contributed by memory-based data structures of dynamic programming that stores the updated data after each iteration. To check whether a non-zero digit is removable during each iteration, the proposed algorithm utilized the most updated collection of non-zero digits from the previous iteration, which ensures that each iteration returns the updated optimal collection.

V. CONCLUSION

In this paper, a new algorithm which minimizes the number of non-zero digits in FIR filter coefficients is proposed. Its minimization process is contributed by itemizing non-zero digits and defining frequency response constraints to implement the knapsack problem. The optimality is ensured by memory-based data structures based on dynamic programming. The proposed algorithm designed the filters with an average of 14.49% less full adder cost over the designs by MCMAT and an average of 47.73% over the designs by MILP. This shows a significant improvement in the complexity of the filter coefficients synthesized by the proposed algorithm for more area-efficient FIR filter implementation.

ACKOWLEDGEMENT

The research and the material reported in this document are supported by the SUTD-MIT International Design Centre (IDC) at Singapore University of Technology and Design.

REFERENCES

[1] F. Feng, J. Chen and C. H. Chang, "Hypergraph Based Minimum Arborescence Algorithm for the Optimization and Reoptimization of Multiple Constant Multiplications," *IEEE Trans. Circuits Syst. I*, vol. 63 , no. 2, Feb. 2016. (to appear)

[2] J. Chen, C. H. Chang, and H. Qian, “New power index model for switching power analysis from adder graph of FIR filter,” in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 2197-2200, Taipei, Taiwan, May 2009.

[3] J. Chen, C. H. Chang, F. Feng, W. Ding and J. Ding, “Novel design algorithm for low complexity programmable FIR filters based on extended double base number system,” *IEEE Trans. Circuits Syst. I*, vol. 62, no. 1, pp. 224–233, Jan. 2015.

[4] S. F. Hsiao, J. H. Zhang Jian and M. C. Chen , "Low-cost FIR filter designs based on faithfully rounded truncated multiple constant multiplication/accumulation," *IEEE Trans. Circuits Syst. II,* vol. 60, no. 5, pp.287 -291 , May. 2013.

[5] Y. J. Yu and Y. C. Lim ,"Design of Linear Phase FIR Filters in Subexpression Space Using Mixed Integer Linear Programming," *IEEE Trans. Circuits Syst. I,*,vol. 54,no.10, pp. 2330-2338 , Oct.2007.

[6] M. Aktan, A. Yurdakul, and G. Dündar, “An algorithm for the design of low-power hardware-efficient fir filters,” *IEEE Trans. Circuits Syst. I*, vol. 55 , no. 7 , pp. 1536–1545, Jul. 2008.

[7] J. Laskowski and H. Samueli, “A 150-MHz 43-Tap Half-Band FIR Digital Filter in 1.2-μm CMOS Generated by Silicon Compiler,” in *Proc. Custom Integrated Circuits Conf.,* pp. 11.4.1-11.4.4, Piscataway, N.J., May1992.

[8] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, “Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination,” *IEEE Trans. CAD*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[9] A. Laming, M. Khemakhem and H. Chabchoub, “Knapsack Problems involving dimensions, demands and multiple choice constraints: generalization and transformations between formulations”, *International Journal of Advanced Science and Technology*, vol. 46, pp. 71-96, Sept. 2012

[10] J. H. McClellan and T. W. Parks, “A personal history of the Parks-McClellan algorithm,” *IEEE Signal Processing Magazine*, vol. 22, no. 2, pp. 82 – 86, Mar. 2005.

[11] C. L. Chen and A. N. Wilson, Jr., “A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 29–39, Jan. 1999.