Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 497471, 9 pages
http://dx.doi.org/10.1155/2014/497471

Research Article

Minimizing Design Costs of an FIR Filter Using a Novel Coefficient Optimization Algorithm

Ming-Chih Chen and Tsung-Ting Chen

Department of Electronic Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung 824, Taiwan

Correspondence should be addressed to Ming-Chih Chen; mjchen@nkfust.edu.tw

Received 26 June 2014; Accepted 19 August 2014; Published 11 September 2014

Academic Editor: Teen-Hang Meen

This work presents a novel coefficient optimization algorithm to reduce the area and improve the performance of finite impulse response (FIR) filter designs. Two basic architectures are commonly used in filters—direct and transposed. The coefficients of a filter can be encoded in the fewest possible nonzero bits using canonic signed digit (CSD) expressions. The proposed optimization algorithm can share common subexpressions (CS) and reduce the number of replicate operations that involve the CSD coefficients of filters with a transposed architecture. The effectiveness of the algorithm is confirmed by using filters with the collision detection multiple access (CDMA) standard, the 121-tap high-pass band, and 105- and 325-tap low-pass bands as benchmarks. For example, the proposed algorithm used in the optimization of 105-tap filter has a 30.44% smaller combinational logic area and a 16.69% better throughput/area than those of the best design that has been developed to date. Experimental results reveal that the proposed algorithm outperforms earlier designs.

1. Introduction

Digital filters have a wide range of applications because they are much more stably reliable than analog filter. Digital filters are used in image/audio processing and a wide range of wired and wireless communication systems. The designs of digital filters vary widely for various applications. They can be divided into finite impulse response (FIR) and infinite impulse response (IIR) filters.

A finite impulse response filter (or so called FIR filter) has a linear phase and arbitrary amplitude, and it is easily implemented. The main goal of the previous designs has been to prevent for a high-cost multiplier at the transmitting side, since a multiplier must be used at the receiving side. The design approach herein involves simplifying the digital filter's coefficients to reduce the area cost of the filter. The coefficients of a digital filter can be separated into various coefficient groups. The filter hardware comprises logical adders, subtractors, and shift registers. If the number of these logical components can be reduced by some simplifying methods, then the overall system can be improved.

The rest of this paper is organized as follows. Section 2 briefly describes previous researches for filter optimization. Section 3 then describes the coefficient optimization method. Next, Section 4 summarizes the experimental results and compares them with those of other previous designs. Conclusions are finally drawn in Section 5, along with recommendations for future research.

2. Related Works

2.1. Coefficient Simplification Methods. Coefficient simplification is one of the most effective ways of improving the area and performance of a finite impulse response filter. Numerous methods of coefficient simplification for filters have been developed. The minimum number of signed power-of-two (MNSPT) methods [1] has been developed to simply numbers of the coefficients. The canonic signed digits (CSD) [2] representation is used to reduce the numbers of binary "1"s in the coefficients to reduce the area of realizing constant multiplications. Simplification algorithms are utilized to reduce the numbers of required constant multipliers in FIR filter

$H0$	0	1	0	1	0	0
$H1$	1	0	1	0	0	0

Figure 1: Two coefficients, $H0$ and $H1$, have horizontal relationships with each other.

$H0$	0	1	0	1	0	0
$H1$	1	0	1	0	0	0
$H2$	1	1	1	1	0	0

Figure 2: Three coefficients, $H0$, $H1$, and $H2$, have vertical relationships with each other.

realization. Using an algorithm to determine the relationships among coefficients and to extract the common terms in their binary formats can reduce the number of redundant logical operations.

In the literature, horizontal and vertical relationships can be found between coefficients; existing relationships can be checked to design an algorithm for extracting their common factors. Such an algorithm commonly has low complexity. The algorithm of Paško et al. [3] performs a global search but consumes too much time. In some studies [4–9], horizontal and vertical relationships between coefficients and the displacement and delay characteristics of coefficients were used to perform the simplifications. Ernesto and Dolecek [10] utilized linear programming to identify the largest common factors of coefficients. Searches for common factors using low-complexity methods can be divided into two categories: horizontal and vertical.

2.1.1. Horizontal Search Algorithms. Horizontal search algorithms find shifting relationships between coefficients. For example, in Figure 1, the coefficients $H0$ and $H1$ in binary format are shifted relative to each other, so they have the same multiplication block. They can both be multiplied by performing only one calculation.

2.1.2. Vertical Search Algorithms. Vertical search algorithms find the delay relationships between the values in corresponding positions of binary representations of coefficients. Coefficients with such a relationship have the same addition block. They can be added by performing a single calculation. Figure 2 presents the vertical search.

2.2. Literature Review. Coefficient representations can be categorized into binary and canonic signed digit (CSD) representations. The CSD representation primarily involves reducing the number of "1"s in the original binary representation of coefficients. More "1"s result in more repeated additions and require more adders in the corresponding circuit realizations. Additionally, repeated additions can be represented as a sequence of additions with a large value and one subtraction. For instance, in binary, the value seven can be expressed as $(2^2 + 2^1 + 2^0)$; it can also be expressed as $(2^3 - 2^0)$ in standard CSD notation. The value seven in CSD notation uses a single subtraction instead of the two additions that are required using binary notation. The cost of realization is reduced by using the CSD notation. Numerous coefficient simplification algorithms are described below.

In 1999, Paško et al. [3] proposed the representation of coefficients using the CSD notation and utilized the horizontal search algorithm to find the common factors (1, 0, 1), (1, 0, −1), (1, 0, 0, 1), (1, 0, 0, −1), and so on. The algorithm did not consider the inversion relationship between pairs of coefficients. The most frequently occurring CSD-based common factor is the one extracted from coefficients. The algorithm is performed until no common factor of coefficients can be found. In 2002, Jang and Yang [4] proposed the representation of coefficients using CSD notation and used a vertical search algorithm to find the common factors (1, −1) and (−1, 1). The algorithm considered terms that were related by inversion. In 2003, Vinod et al. [5] utilized the vertical common term extraction method that was developed by Jang and Yang [4] to simplify coefficients. Their method firstly performs horizontal searches for common factors (1, 0, 1), (1, 0, −1), (1, 0, 0, 1), and (1, 0, 0, −1) and then vertical searches for factors (1, 0, 1) and (−1, 0) related by inversion.

In 2005, Vinod and Lai [11] improved the horizontal and vertical search algorithms that were developed in 2003 [5]. They constructed multiplier block adders (MBAs) and then structure adders (SAs). Their new algorithm yielded a final result after adding one or more delays to the logic gates of realized noncommon factors of the coefficients. The algorithm that was presented by Takahashi and Yokoyama [6] extracts common factors by finding the common factor with the highest frequency. If two or more common factors have the same frequency, then the smallest one is extracted. The experience of performing the algorithm in a filter with 26 coefficients shows that the factors (1, 0, N) and (1, 0, 0, N) are found to appear most frequently. Maskell and Liewo [12] developed an algorithm for reducing the height *in all instances* of an adder tree that is composed of common factors. The height of the adder tree can be reduced by properly setting the width of the adder. Accordingly, the common factors that are extracted more resulted in the less wide adders with low latency and low area cost. A local search algorithm firstly extracts the common factors (1, 0, 1) and (1, 0, −1). The algorithm also uses a specific multiplier block (MB) in place of a full adder (FA), reducing the area cost by 67%.

In 2007, Smitha and Vinod [7] proposed the use of binary representation to extract common factors of coefficients with only two- to four-bit terms. The algorithm uses as few adders to generate the coefficients as possible and performs a horizontal search to find the common factors (1, 1), (1, 0, 1), (1, 1, 1), (1, 1, 1, 1), (1, 1, 0, 1), (1, 0, 1, 1), and (1, 0, 0, 1). In 2010, Vinod et al. [9] developed a new algorithm to perform a horizontal search by extracting the common factors (1, 0, 1), (1, 0, −1), (1, 0, 0, 1), and (1, 0, 0, −1) of the coefficients in CSD notation. Then, the algorithm performs a vertical search to extract the factors (1, 1) and (1, 0, 1). The horizontal search part of this algorithm takes into account the inversion of the filter's coefficients. In previous works, the search algorithms effectively reduced the number of logical operations required for multiplication by a constant by extracting the common factors. The present work proposes a new search algorithm

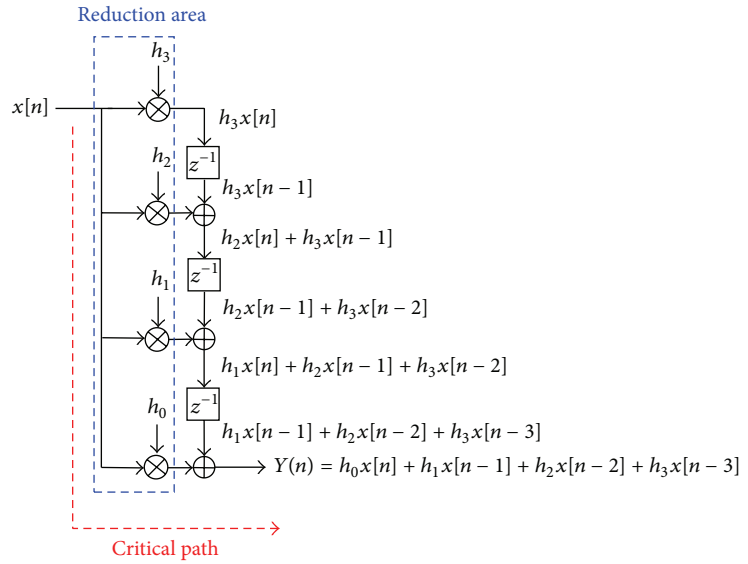

Figure 3: An FIR filter of the transposed form.

```
Define FilterTap;
Max = FilterTap-1;
//Set the terminated condition.
While(true){
   N = 0; MaxSE_Cnt = 0; //Initialize Variable.
   While (N <= Max){
      RecordTable ← Count_SE(Filter Coefficient); //Step 1
      N = N + 1; //Step 2
   }
   RecordTable ← InverseCount(RecordTable);
   //Step 3
   For (i = 0; i < RecordTable.size; i++)
      MaxSE_Cnt = Max(RecordTable[i].Counts, MaxSE_Cnt); //Step 4
   If (MaxSE_Cnt > 1) //Step 5
      Filter Coefficient ← SimplifyCoefficient(Filter Coefficient);
   Else
      Break; //Step 5 terminated
}
```

Algorithm 1: Proposed CCSE algorithm.

to improve the area cost and the performance over those of earlier designs.

3. Proposed Coefficient Optimization Method

Equation (1) is the finite impulse response (FIR) filter. At various time points k, the variable x is calculated only with the coefficient h. Consider the following:

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k). \tag{1}$$

The basis of the common subexpression elimination (CSE) algorithm is to find the common factors of the coefficients of a filter. In the transposed form, presented in Figure 3, the common factors of the coefficients are evaluated as the shared multiplication blocks (MBs). Therefore, the overall area of the FIR circuit can be reduced by the sharing of MBs.

This section elucidates the use of a new CSE algorithm to extract the common factors of the coefficients in CSD notation. The algorithm obtains the statistics concerning the frequencies of the appearances of coefficients and finds the reciprocals of those coefficients to search for common factors. Algorithm 1 presents the pseudocode of the proposed CCSE (CSD-based common subexpression elimination) algorithm. The steps of the proposed CCSE algorithm are as follows.

Initial. Set the boundary conditions $0 \leqq N \leqq$ (filter's order -1) and initialize N to zero.

Step 1. Find the Nth coefficient and find all nonzero bit positions (from high to low) of the coefficient. Record these positions and list the combinations of subexpressions (SEs) with more than one nonzero bit. Use all of the combinations as the basic elements (BEs) in simplification (tabulate the BEs). If an input element matches one of the BEs in the table, then increase the statistical frequency of the BE. Otherwise, if an input element does not match any BE in the table, the input element becomes a new BE and is added to the table.

Step 2. Set $N = N + 1$ and determine whether the value N exceeds the value in the boundary condition set in the initialization. If it does not, then repeat Step 1; otherwise, proceed to Step 3.

Step 3. Evaluate all of the BEs in the table to find their subexpressions (SEs) having reciprocal SEs. If the inverted SEs exist, then use the positive SEs as basic elements. Calculate the number of appearances of negative SEs.

Step 4. Evaluate all of the BEs in the table to find which BE has the highest appearance frequency. If the highest frequency is one, then the algorithm proceeds to the final step; if the highest frequency exceeds one, then select the BEs as common subexpressions (CSs), and if more than one BE has the same highest frequency and this frequency exceeds one, select the shorter BE as the extracted CS.

Step 5. Find all of the coefficients with the same CS that was generated in Step 4 and those of the corresponding inverted CS and perform the elimination process. When the process is complete, put a new replaced variable back in the original expressions and reset the loop value N to zero, before returning to Step 1.

Final. Complete the algorithm and output the simplification results.

Table 10 presents the example of a filter with three coefficients to elucidate the actual processes of the algorithm. Intermediate results are obtained after each step of the algorithm, as described in the following statements.

Initial. Set $N = 0$ and the boundary condition $0 \leqq N \leqq 2$.

Explanation 1 (N equals zero). Select the coefficient $H(0)$ and list all of the SEs whose nonzero bits are greater than one. Make these SEs to BEs for simplification. The appearance frequencies of these SEs are as follows.

SE (1, 0, 1) appears twice.

SEs (1, 0, 0, 1), (1, 0, 0, 0, 1), (1, 0, 1, 0, 1), (1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 0, 0, 1, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), and (1, 0, 1, 0, 1, 0, 0, 1) all appear once.

Firstly, no BE is recorded in the table, so all SEs are taken as BEs for subsequent simplification.

Explanation 2 ($N = N + 1$). N equals one and satisfies $0 \leqq N \leqq 2$. The algorithm executes Step 1.

Explanation 3 (N equals one). Select the coefficient $H(1)$ and list all of the SEs whose nonzero bits are greater than one. The intermediate results are as follows.

SE (1, 0, 1) appears twice.

SEs (1, 0, 0, 1), (1, 0, 0, 0, 1), (1, 0, 1, 0, 1), (1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 0, 0, 1, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), and (1, 0, 1, 0, 1, 0, 0, 1) appear once.

The BEs in the table are as follows.

(1, 0, 1) appears four times.

(1, 0, 0, 1), (1, 0, 0, 0, 1), (1, 0, 1, 0, 1), (1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 0, 0, 1, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), and (1, 0, 1, 0, 1, 0, 0, 1) all appear twice.

Explanation 4 ($N = N + 1$). N equals two and satisfies $0 \leqq N \leqq 2$. The algorithm executes Step 1.

Explanation 5 (N equals two). Select the coefficient $H(2)$ and list all of the SEs whose nonzero bits are greater than one. The intermediate results are as follows.

SE (-1, 0, -1) appears once.

The BEs in the table are as follows.

(1, 0, 1) appears four times.

(-1, 0, -1), (1, 0, 0, 1), (1, 0, 0, 0, 1), (1, 0, 1, 0, 1), (1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 0, 0, 1, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), and (1, 0, 1, 0, 1, 0, 0, 1) all appear twice.

Explanation 6 ($N = N + 1$). N equals three and so does not fall in the range $0 \leqq N \leqq 2$. The algorithm goes to Step 3.

Explanation 7. Determine whether the CSs of the BEs in the table have corresponding inverted CSs. SEs (-1, 0, -1) and (1, 0, 1) are the inverse of each other. The SE (1, 0, 1) becomes the basic element. The number of appearances of SE (-1, 0, -1) is calculated.

The BEs in the table are as follows.

(1, 0, 1) appears five times.

(1, 0, 0, 1), (1, 0, 0, 0, 1), (1, 0, 1, 0, 1), (1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 0, 0, 1, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), and (1, 0, 1, 0, 1, 0, 0, 1) all appear twice.

Explanation 8. Make the BE with the highest frequency in the CS to be simplified. BE (1, 0, 1) is selected as the CS after Step 4 is performed.

Explanation 9. The CS (1, 0, 1) is taken from Step 4 and the inverse CS (-1, 0, -1) is also utilized in the simplification. A new variable replaces the CS in the original coefficients and the intermediate outputs are as presented in Table 11. After the five steps have been completed, the algorithm resets the value N to zero and returns to Step 1.

Explanation 10. Repeat the five steps and find the BEs of coefficients until the appearance frequencies of BEs are equal to zero. Table 12 presents the outputs of the algorithm.

The algorithm yields the following CSs.

C5 = (1, 0, 1), whose decimal value is 5;

C21 = (5, 0, 1), whose decimal value is 21;

C169 = (21, 0, 0, 1), whose decimal value is 169.

Table 1: Performance comparisons of filter with 48 tap in CDMA 2000 [13]; bit width of the filter is 16 bits.

Algorithm	Combinational unit (um^2)	Buf/Inv unit (um^2)	Noncombinational unit (um^2)	Total area (um^2)
Origin CSD	86925.49 (0%)	2122.24	49456.91	136382.40 (0%)
Paško et al. [3]	83825.28 (−3.57%)	1729.73	49456.91	133282.19 (−2.27%)
Jang and Yang [4]	83326.32 (−4.14%)	1782.95	50574.59	133900.90 (−1.81%)
Vinod et al. 2003 [5]	84520.50 (−2.77%)	1762.99	49456.92	133977.41 (−1.76%)
Vinod et al. 2010 [9]	83848.57 (−3.54%)	1683.16	50574.59	134423.15 (−1.44%)
Our CCSE	75253.15 (−13.43%)	1270.68	49456.92	124710.06 (−8.56%)

Based on the above explanations and the pseudocode of the proposed algorithm, the main goal of the first step is to identify all subexpressions with nonzero bits. An algorithm that finds more subexpressions is more likely to find the best simplification. The third step is to calculate the number of inverted SEs, with a view to improving the area of simplification. The fourth and fifth steps are the major reduction steps. The major difference between the proposed algorithm and earlier ones concerns the CSE processes. The proposed algorithm does not directly eliminate the CSs from the coefficients but replaces the CSs with new variables. In next iteration, the algorithm performs the extraction of new CS between the coefficients with new variables and those without CS. This new approach can extract more new CSs and achieve better simplification results.

To confirm the effectiveness of the proposed algorithm, the filter with three taps is utilized to estimate the required logical operations at the architectural level. The input variable is set to 12 bits. The minimum bit widths of the coefficients and common factors are utilized in the estimation. A represents the adder; S denotes the subtractor; I denotes the inverter. The realization areas of a filter with original three coefficients in CSD notation are as follows. The realization of a filter with coefficient $H(0)$ needs (72 A, 0 S, 0 I), $H(1)$ needs areas of (60 A, 0 S, 0 I), and $H(2)$ needs areas of (0 A, 30 S, 0 I). The total area of the filter needs (132 A, 30 S, 0 I). After the algorithm is implemented, the realization of common subexpression C5 needs (15 A, 0 S, 0 I), C21 needs (17 A, 0 S, 0 I), and C169 needs (20 A, 0 S, 0 I). $H(0)$ and $H(1)$ share the common subexpression C169 with a shifting relationship. The shift relationship can be realized without occupying additional area. $H(2)$ has the inverted C5. The realization of $H(2)$ only needs (15 A, 0 S, 15 I) after the algorithm is executed. The total area cost of the filter is (52 A, 0 S, 15 I). In the estimation of the area, the subtractor and the adder are assumed to have the same area cost. The area cost of the inverter is 1/10 of that of the adder. Accordingly, the original area cost of the filter is 162 A. The proposed algorithm reduces the area cost of the filter to 53.5 A. The algorithm reduces 67% of the area cost of realizing the coefficients of the filter.

4. Experimental Results

In the experiments herein, four filters are used to compare the performance of various search algorithms; the filters are a symmetric filter with 48 tap defined in CDMA 2000 communication protocol, a high-pass filter with 121 tap, and two low-pass filter with 105 and 325 taps. The CDMA 2000 [13] is a 3 G mobile communication standard. The 3 G system offers various telecommunications services, including voice, multimedia, and high-speed and low-speed data transmissions. The system requires a based band filter to perform intersymbol interference. The CDMA 2000 standard recommends the use of a symmetric finite impulse response filter with 48 tap to eliminate the interference. A high-pass filter with 121 tap [14] and a low-pass filter with 105 and 325 taps [15] are also used to confirm the effectiveness of simplification by the search algorithms. Symmetric coefficients of the three filters are realized with the transposed architectures.

In the realization, the Synopsys Design Compiler (DC) SP1 software is utilized to data concerning the synthesis of the circuit. The process technology adopts the CBDK Arm 4.0_TSMC 0.18 um cell library with the default system parameters. The following data in the compared table are rounded to the second decimal place, the unit of circuit area is um^2, the unit of data arrival time is nanosecond (ns), the unit of the throughput is Gigabits per second (Gbps), and the unit of throughput per area is bps/um^2. In the comparison of areas, combination logic area is utilized to confirm the simplification performance of each algorithm. The search algorithms can simplify the coefficients, which are realized using combinational logic units. In the following performance comparisons, the coefficients with original CSD-based expressions are only simplified using the Synthesis DC tool. Other search algorithms perform simplifications of the coefficients in CSD notation.

According to Table 1, the search algorithm proposed by Jang and Yang [4] has a coefficient simplification ratio of 4.14% compared with original filter, which is better than that of any previous algorithm. The algorithm extracts the most common factors than the others but also causes the most path delays. The proposed CCSE algorithm reduces the combination logic area by 13.43%, and the total filter area by 8.56% compared with original filter. The proposed algorithm reduces the area of the filter by more than the previous algorithms. According to Table 2, the search algorithm that was developed by Jang and Yang [4] has the best simplification ratio, 15.85%, which corresponds to the best reduction of the area of coefficient realization. The proposed algorithm reduces the combinational logic area by 22.91% and the total filter area by 15.57%. Both of these results are the best achieved using any algorithm. Tables 3 and 4 also reveal that

Table 2: Performance comparisons of high-pass filter [14] with 121 tap; bit width of filter is 16 bits.

Algorithm	Combinational unit (um^2)	Buf/Inv unit (um^2)	Noncombinational unit (um^2)	Total area (um^2)
Original CSD	282012.19 (0%)	6167.15	132932.92	414945.12 (0%)
Paško et al. [3]	237312.03 (−15.85%)	2561.33	132932.92	370244.95 (−10.77%)
Jang and Yang [4]	258341.53 (−8.39%)	5348.85	133771.17	392112.71 (−5.50%)
Vinod et al. 2003 [5]	243954.85 (−13.50%)	4390.85	135168.26	379123.12 (−8.63%)
Vinod et al. 2010 [9]	240861.30 (−14.59%)	4577.13	135168.27	376029.56 (−9.38%)
Our CCSE	217403.53 (−22.91%)	3419.54	132932.92	350336.45 (−15.57%)

Table 3: Performance comparisons of low-pass filter [15] with 105 tap; bit width of filter is 16 bits.

Algorithm	Combinational unit (um^2)	Buf/Inv Unit (um^2)	Noncombinational unit (um^2)	Total area (um^2)
Original CSD	307562.27 (0%)	7457.78	114980.34	422542.61 (0%)
Paško et al. [3]	244680.00 (−20.45%)	3056.96	114980.34	359660.35 (−14.88%)
Jang and Yang [4]	264545.26 (−13.99%)	7118.49	116098.01	380643.28 (−9.92%)
Vinod et al. 2003 [5]	251242.99 (−18.31%)	6443.23	117215.68	368458.67 (−12.80%)
Vinod et al. 2010 [9]	245312.02 (−20.24%)	6233.67	116098.01	361410.03 (−14.47%)
Our CCSE	213947.39 (−30.44%)	4367.56	114980.34	328927.74 (−22.16%)

the proposed algorithm reduces the area of the filter more than does any other.

As more common factors are extracted using the search algorithms, the path delay (or the data arrival time) is increased more. The realizations under the coefficients in CSD notation have shorter path delays compared to those of the coefficients in original binary notation after search algorithms are implemented. The throughputs have the same effects as the path delays. In Table 5, all of the throughput/area ratios are reduced by realizing the constant multiplications with the coefficients in CSD notation. The proposed algorithm increases the throughput/area ratio by 2.84%, which is the largest increase of any algorithm. In Table 7, the coefficient simplifications using the proposed algorithm increase the throughput/area ratio by 16.69% more than the original coefficient simplifications using Synopsys DC. Tables 6 and 8 also reveal that the proposed algorithm has the best simplification ratio of any of the compared algorithms.

Previously proposed search algorithms have different advantages in coefficient simplifications than those of the four filters described above. All of the algorithms effectively reduce the area cost of realizing the FIR filters; most of them sacrifice throughput by increasing the path delays. The proposed CCSE algorithm reduces the most area of the filter, but it has a higher throughout than the other algorithms. According to experimental observations, the proposed algorithm performs best not only in area reduction but also in the throughput/area ratio.

However, the layout realizations of 48-, 121-, 105-, and 325-tap filters are also utilized to reveal the effectiveness of our proposed algorithm. The Cadence SOC Encounter software is utilized to place and route the designed filters. The I/O pin counts of all filters are 40 pins. The synthesis and layout processes utilize TSMC 0.18 μm mixed-signal RF 1P6 M CMOS technology. Table 9 shows the placement and route information of four designed filters. The die size of 48-tap CDMA filter is about 0.16 mm^2 and the total gate count of the filter is approximately 12.5 K. The filter can operate at 72 MHz with 7.531 mW power dissipation. Figure 4 shows the layout graph of the 48-tap filter with 40 I/O pins. The 325-tap filter has the largest chip size and can operate at 55 MHz with an area of 87.68 K gates.

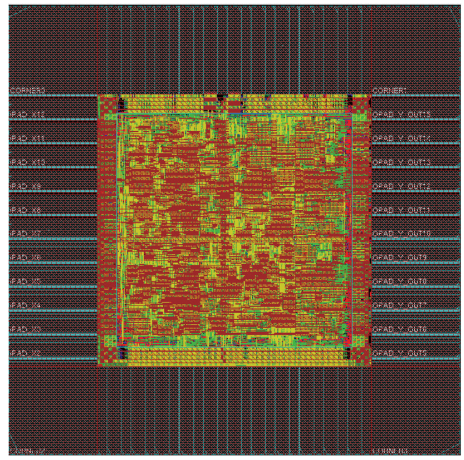

Figure 4: Layout graph of 48-tap CDMA filter.

5. Conclusions

In summary, coefficient simplifications by the search algorithms are useful in reducing the combinational logic area

Table 4: Performance comparisons of low-pass filter [15] with 325 tap; bit width of filter is 16 bits.

Algorithm	Combinational unit (um^2)	Buf/Inv unit (um^2)	Noncombinational unit (um^2)	Total area (um^2)
Origin-CSD	672285.40 (0%)	17892.70	361007.54	1033292.94 (0%)
Paško et al. [3]	584887.56 (−13.00%)	6646.14	361007.54	945895.11 (−8.46%)
Jang and Yang [4]	653404.75 (−2.81%)	16788.34	361985.50	1015390.26 (−1.73%)
Vinod et al. 2003 [5]	588663.03 (−12.44%)	12716.82	363242.88	951905.91 (−7.88%)
Vinod et al. 2010 [9]	581780.71 (−13.46%)	12301.02	363242.88	945023.59 (−8.54%)
Our CCSE	514031.92 (−23.54%)	8286.06	361007.54	875039.46 (−15.32%)

Table 5: Performance comparisons of filter with 48 tap in CDMA 2000 [13]; bit width of filter is 16 bits.

Algorithm	Data arrival time (ns)	Throughput (Gbps)	Throughput/total area (bps/um^2)
Original CSD	10.58	1.51	11071.80 (0%)
Paško et al. [3]	11.14	1.44	10804.14 (−2.42%)
Jang and Yang [4]	11.45	1.40	10455.49 (−5.57%)
Vinod et al. 2003 [5]	11.62	1.38	10300.24 (−6.97%)
Vinod et al. 2010 [9]	11.31	1.41	10489.26 (−5.26%)
Our CCSE	11.28	1.42	11386.41 (+2.84%)

Table 6: Performance comparisons of high-pass filter [14] with 121 tap; bit width of filter is 16 bits.

Algorithm	Data arrival time (ns)	Throughput (Gbps)	Throughput/total area (bps/um^2)
Original CSD	12.30	1.30	3132.94 (0%)
Paško et al. [3]	13.54	1.18	3187.08 (+1.73%)
Jang and Yang [4]	14.76	1.08	2754.31 (−12.09%)
Vinod et al. 2003 [5]	13.99	1.14	3006.94 (−4.02%)
Vinod et al. 2010 [9]	13.43	1.19	3164.64 (+1.01%)
Our CCSE	14.28	1.12	3196.93 (+2.04%)

Table 7: Performance comparisons of low-pass filter [15] with 105 tap; bit width of filter is 16 bits.

Algorithm	Data arrival time (ns)	Throughput (Gbps)	Throughput/total area (bps/um^2)
Original CSD	12.19	1.31	3100.28 (0%)
Paško et al. [3]	14.21	1.13	3141.85 (+1.34%)
Jang and Yang [4]	15.78	1.01	2653.40 (−14.41%)
Vinod et al. 2003 [5]	14.36	1.11	2985.41 (−3.71%)
Vinod et al. 2010 [9]	14.04	1.14	3154.31 (+1.74%)
Our CCSE	13.4	1.19	3617.82 (+16.69%)

Table 8: Performance comparisons of low-pass filter [15] with 325 tap; bit-width of filter is 16 bits.

Algorithm	Data arrival time (ns)	Throughput (Gbps)	Throughput/total area (bps/um^2)
Original CSD	12.55	1.27	1229.08 (0%)
Paško et al. [3]	15.21	1.05	1110.06 (−9.78%)
Jang and Yang [4]	16.22	0.99	974.99 (−20.67%)
Vinod et al. 2003 [5]	15.74	1.02	1071.53 (−12.82%)
Vinod et al. 2010 [9]	15.89	1.01	1068.76 (−13.04%)
Our CCSE	14.62	1.09	1245.66 (+1.35%)

Table 9: Placement and route information of four designed filters.

Filters	48 tap	121 tap	105 tap	325 tap
Die size	$400 \times 400\ \mu m^2$	$710 \times 710\ \mu m^2$	$720 \times 720\ \mu m^2$	$1400 \times 1400\ \mu m^2$
Frequency	72 MHz	66 MHz	68 MHz	55 MHz
Gate count	12,495 gates	35,104 gates	32,989 gates	87,680 gates
Fault coverage	99.3% with 40 test patterns	99.4% with 40 test patterns	99.5% with 40 test patterns	99.1% with 40 test patterns
Code coverage	100% with 256 test patterns	100% with 256 test patterns	100% with 256 test patterns	100% with 256 test patterns
Operate voltage	1.8 V	1.8 V	1.8 V	1.8 V
Power dissipation	7.531 mW @ 72 MHz	20.3 mW @ 66 MHz	21.4 mW @ 68 MHz	52.8 mW @ 55 MHz

Table 10: Coefficients of a filter with three taps.

$H(0)$	1	0	1	0	1	0	0	1	0	0	0	0
$H(1)$	0	0	0	0	1	0	1	0	1	0	0	1
$H(2)$	0	0	0	0	0	0	0	0	0	–1	0	–1

Table 11: Intermediate results obtained after performing five steps of the proposed algorithm.

$H(0)$	0	0	***5***	0	1	0	0	1	0	0	0	0
$H(1)$	0	0	0	0	0	0	***5***	0	1	0	0	1
$H(2)$	0	0	0	0	0	0	0	0	0	0	0	***–5***

Table 12: Outputs of algorithm.

$H(0)$	0	0	0	0	0	0	0	***169***	0	0	0	0
$H(1)$	0	0	0	0	0	0	0	0	0	0	0	***169***
$H(2)$	0	0	0	0	0	0	0	0	0	0	0	***–5***

and the total filter area. Previously developed search algorithms provide greater area reductions but they sacrifice the throughput/area ratio. The proposed CCSE algorithm offers the best area reduction and the best throughput/area ratio. The main feature of the proposed algorithm is that the coefficients are not eliminated after they are used to reduce the number of common factors; instead, they are retained to reduce the number of new common factors that are generated by combining existing common factors to maximize the area reduction. Experimental results further reveal that the proposed algorithm performs best.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the Ministry of Education of the Republic of China, Taiwan, for financially supporting this research under Contract no. C302-05. Ted Knoy is appreciated for his editorial assistance.

References

[1] Y. C. Lim, J. B. Evans, and B. Liu, "Decomposition of binary integers into signed power-of-two terms," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 6, pp. 667–672, 1991.

[2] R. M. Hewlitt and E. S. Swartzlander Jr., "Canonical signed digit representation for FIR digital filters," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS '00)*, pp. 416–426, October 2000.

[3] R. Paško, P. Schaumont, V. Derudder, S. Vernalde, and D. Ďuračková, "A new algorithm for elimination of common subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 1, pp. 58–68, 1999.

[4] Y. Jang and S. Yang, "Low-power CSD linear phase FIR filter structure using vertical common sub-expression," *Electronics Letters*, vol. 38, no. 15, pp. 777–779, 2002.

[5] A. P. Vinod, E. M.-K. Lai, A. B. Premkumar, and C. T. Lau, "FIR filter implementation by efficient sharing of horizontal and vertical common subexpressions," *Electronics Letters*, vol. 39, no. 2, pp. 251–253, 2003.

[6] Y. Takahashi and M. Yokoyama, "New cost-effective VLSI implementation of multiplierless FIR filter using common subexpression elimination," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 1445–1448, May 2005.

[7] K. G. Smitha and A. P. Vinod, "A new binary common subexpression elimination method for implementing low complexity FIR filters," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 2327–2330, May 2007.

[8] R. Mahesh and A. P. Vinod, "A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 217–229, 2008.

[9] A. P. Vinod, E. Lai, D. L. Maskell, and P. K. Meher, "An improved common subexpression elimination method for reducing logic operators in FIR filter implementations without increasing logic depth," *Integration, the VLSI Journal*, vol. 43, no. 1, pp. 124–135, 2010.

[10] D. T. E. Ernesto and G. J. Dolecek, "Novel multiplierless FPGA implementation of CDMA 2000 baseband filter," in *Proceedings of the International Symposium on Communications and Information Technologies (ISCIT '08)*, pp. 289–294, October 2008.

[11] A. P. Vinod and E. M.-K. Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 295–304, 2005.

[12] D. L. Maskell and J. Liewo, "Hardware-efficient FIR filters with reduced adder step," *Electronics Letters*, vol. 41, no. 22, pp. 1211–1213, 2005.

[13] G. L. Do and K. Feher, "Efficient filter design for IS-95 CDMA systems," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 4, pp. 1011–1020, 1996.

[14] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Transactions on Circuits and Systems*, vol. 30, no. 10, pp. 723–739, 1983.

[15] A. Shahein, Q. Zhang, N. Lotze, and Y. Manoli, "A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 3, pp. 616–627, 2012.