ASSUMEs: Heuristic Algorithms for Optimization of Area and Delay in Digital Filter Synthesis

Levent Aksoy
Istanbul Technical University
Istanbul, Turkey

Eduardo Costa
Universidade Catolica de Pelotas
Pelotas, Brazil

Paulo Flores, Jose Monteiro
INESC-ID/IST
Lisbon, Portugal

***Abstract*—In this work two heuristic algorithms are presented for the problems of optimization of area and optimization of area under a delay constraint in digital filter synthesis. The heuristics search for a solution on a combinational network that represents a covering problem using a greedy method for partial term selection, starting, for each coefficient, from the outputs towards the inputs. This top-down approach considers a much larger solution space than existing bottom-up heuristic algorithms. We present results on a wide range of instances and compare them with exact and prominent heuristic algorithms. The results demonstrate that the solutions obtained by the proposed heuristics are extremely close to the exact solutions and are significantly better than the existing heuristic algorithms.**

I. INTRODUCTION

Finite impulse response (FIR) digital filters are widely used in digital signal processing by virtue of stability and easy implementation. The problem of designing FIR filters has received a significant amount of attention during the last decade, as the filters require a large number of multiplications, leading to excessive area, delay, and power consumption even if implemented in a full custom integrated circuit. Previous works [1]-[7] have focused on the design of filters with minimum area by replacing the multiplication operations with constant coefficients by addition, subtraction, and shifting operations. Since shifts are free in terms of hardware, the design problem can be defined as the minimization of the number of addition/subtraction operations to implement the coefficient multiplications. In fact, this is known, more generally, as the Multiple Constant Multiplications (MCM) problem and has applications on a wide range of problems, e.g., [1] and [2]. In this paper, we focus on FIR filter implementations.

To further reduce the complexity of the design, the coefficients can be expressed in canonical sign digit (CSD) or represented in minimal sign digit (MSD). The CSD representation is a signed digit system with the digit set {-1,0,1}. The CSD representation is unique and has two main properties: i) the number of non-zero digits is minimal, ii) two non-zero digits are not adjacent. This representation is widely used in multiplier-less implementations because it reduces the number of non-zero digits by 33% on average compared with the binary representation. The MSD representation is obtained by dropping the second property of the CSD representation. Thus, a constant can have several MSD representations, but all with a minimum number of non-zero digits. For example, the value 6 is represented using 4 digits in CSD as 10(-1)0 but both 10(-1)0 and 0110 are valid representations in MSD. In MCM, it is more efficient to use MSD representation that has the same number of non-zero digits as CSD but provides multiple alternative representations for a constant coefficient [3].

Recent works [8]-[11] deal with the problem of optimization of area and delay simultaneously. The problem can be described as the minimization of the number of operations such that a user-specified delay is not exceeded. As the delay is dependent on several implementation issues, such as circuit technology and routing, in this paper, we define the delay as the number of adder-steps, which denotes the maximal number of adders/subtracters in series to produce any multiplication as given in [8]. Since the definition of adder-steps is identical to the definition of levels in combinational circuits, we use both definitions interchangeably in this paper.

The algorithms introduced in this paper optimize area and delay in the design of digital filters. The heuristics search for a solution on the combinational network generated by the exact algorithm of [4], representing the covering problem to be solved. Instead of solving the minimum cover problem as done in [4] and [10], a greedy method is used to synthesize each coefficient by considering not-yet synthesized coefficients. Hence, the solutions of the filter instances that are hard to be found by the exact algorithms are obtained easily. Besides, since the proposed heuristic algorithms consider more possible implementations of a coefficient, they search a solution in a larger space than the other existing heuristic algorithms that find common subexpressions, e.g., [1], [2], and [9].

The paper is organized as follows. In Section II, the related work is presented. Afterwards, the proposed heuristic algorithms are introduced. Experimental results are given in Section IV. Finally, the paper concludes in Section V.

II. RELATED WORK

In this section, initially, we describe the exact and heuristic algorithms previously proposed for the problems of optimization of area and optimization of area under a delay constraint. Then, we present the main differences between our heuristic algorithms and the other algorithms.

A. Exact Algorithms

An exact algorithm for the maximal sharing of the partial terms for more than one coefficient is given in [4]. This algorithm can handle binary, CSD, and MSD representations of the coefficients. In this algorithm, the filter design problem is defined as a binate

covering problem, a special case of a 0-1 integer linear programming (ILP) problem where every constraint is interpreted as a propositional clause. In the preprocessing phase of the algorithm, after the coefficients of the filter are made positive and odd, all possible partial terms that may be used to generate the set of coefficients are found. The partial terms are obtained with addition/subtraction of non-zero digit combinations to the coefficients. As an example, consider 15 (1111, in binary) as a coefficient to be synthesized. The possible partial terms that implement 15 are given in Fig. 1. Observe that since the shifts are free in terms of hardware, the implementation of (8+7) is the same as the implementation of (1+14). So, (1+14) is not listed, as the duplications of implementations, e.g., (9+6) is equal to (6+9), are not listed in this figure. After the partial terms for the coefficient 15, i.e., 3, 5, 7, 9, 11, and 13, are obtained, the partial terms and operations that implement these partial terms are also found in this way. Note that when coefficients are represented in CSD or MSD, both subtraction and addition are performed to find partial terms.

Then, a combinational network that only includes AND and OR gates is constructed by [4]. In this network, an AND gate represents an addition/subtraction operation and an OR gate combines the possible ways of implementation of a partial term. The primary inputs of the network represent the filter input or its shifted versions. The primary outputs of the network are the OR gate outputs that generate the coefficients of the filter. The number of inputs for each AND gate is two: these are either primary inputs or OR gate outputs (partial terms). The inputs of an OR gate are the outputs of AND gates associated with the partial term. The combinational network generated by the algorithm for the coefficient 15 in binary is given in Fig. 2. Observe that the network is a representation of a covering problem.

After the combinational network is generated, additional hardware (with optimization variables) is added to the network and the variables that represent the filter coefficients are assigned to 1. Then, all the conjunctive normal form (CNF) formulas for each gate output are obtained. Each clause in the CNF formula is defined as a constraint by expressing each clause as a linear inequality. Finally, an objective function to be minimized is constructed. The objective function is a linear combination of the optimization variables that are associated with the partial terms in the network. A generic SAT-based 0-1 ILP solver is used to obtain the exact solution.

In [10], an exact algorithm for the minimization of area under a delay constraint problem is proposed where the network is generated as in [4] and it is modified in order to handle the delay constraints. In this algorithm, the AND gates (operations) on each path that exceeds the minimum delay of the network are expressed in a constraint forcing that all these operations must not be found together in the solution. With the additional constraints, the network is given to the 0-1 ILP solver to obtain a minimum area solution.

B. *Heuristic Algorithms*

There have been a number of proposed techniques on the optimization of area of the FIR digital filters. These works are based on finding common digit patterns in the coefficients. These methods range from graph based coefficient synthesis techniques [5], [6] and incorporation of two-term common subexpressions [1] to exhaustive enumeration of all possible digit patterns [7].

With 1 non-zero digit combinations	With 2 non-zero digit combinations
1111=1000+0111=1<<3+7<<0	1111=1100+0011=3<<2+3<<0
1111=0100+1011=1<<2+11<<0	1111=1010+0101=5<<1+5<<0
1111=0010+1101=1<<1+13<<0	1111=1001+0110=9<<0+3<<1

Figure 1. Implementations of the coefficient 15 in binary.

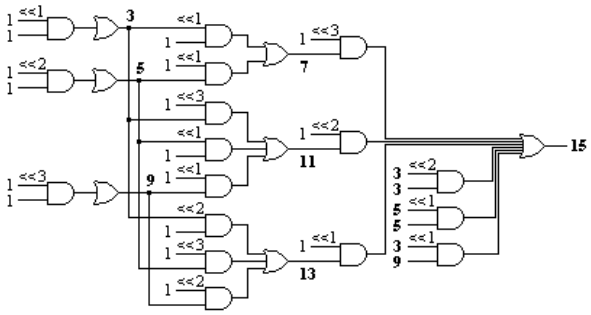

Figure 2. The network generated for the coefficient 15 in binary.

Despite the large number of techniques proposed for optimization of area, there are not many methods that also consider the delay of the design, which is essential for high-speed systems. In [9] and [11], while minimizing area, delay is also considered in the selection criterion of the partial terms. In [8], initially, the number of addition/subtraction operations is reduced and then, a set of transformations in an iterative loop is used to reduce the delay.

C. *Proposed Heuristic Algorithms*

The heuristic algorithms proposed in this paper use the combinational network generated by the exact algorithm presented in [4]. These algorithms synthesize each coefficient one at a time by selecting an operation among the set of possible operations. The selection is done in a greedy manner by considering not-yet synthesized coefficients. Since these algorithms do not attempt to solve the minimum cover problem presented by the network, they may find an optimal solution (local minimum) rather than the minimum (global minimum), but in a reasonable time.

The proposed heuristics consider much more possible implementations of a coefficient than the heuristic algorithms that find pairs of the most common non-zero digits [2] or the two-term common subexpressions [1], [9]. Also, the heuristic algorithm designed for the optimization of area under a delay constraint problem is implemented in a top-down approach (from primary outputs to primary inputs) where it has more possibilities to synthesize a partial term while controlling the delay than the heuristic algorithm that uses a bottom-up approach [9].

III. HEURISTIC ALGORITHMS FOR DIGITAL FILTER SYNTHESIS

In this section, we describe the proposed heuristics briefly. Initially, we present the heuristic called ASSUME-A designed for optimization of area and then, the heuristic called ASSUME-D designed for optimization of area under a delay constraint.

A. *Area Optimization: ASSUME-A Algorithm*

After the network is generated as described in [4], the *min-adder* and *max-level* values of each operation and partial term are found by traversing from primary inputs to primary outputs in the preprocessing phase. The *min-adder* is the minimum number of operations that are required to implement an operation or a partial term. The *min-adder* value of a partial term (OR gate) is determined by finding the minimum of the *min-adder* values of operations (AND gates) that implement the partial term. The *min-adder* value of an operation (AND gate) is the sum of the *min-adder* values of

its inputs plus 1, if the inputs are different; otherwise it is the *min-adder* value of an input plus 1. Note that the *min-adder* value of a primary input is 0. The *max-level* is the maximum number of operations in series that implement the operation or the partial term. Consider the network given in Fig. 2 with the partial term 15. The *min-adder* and *max-level* value of 15 is 2 and 3 respectively.

ASSUME-A has two main parts: minimal and optimal as the heuristic algorithm of [5]. The algorithm is as follows:

1. Store the pre-processed coefficients of the filter (primary outputs of the network, all made positive and odd) in a set called *Aset* and label them as unimplemented.
2. *The minimal part*: For each element labeled as unimplemented in *Aset*, if the element is implemented in the network with an operation whose inputs are primary inputs or in *Aset* then, synthesize the element with the operation and label it as implemented.
3. If there is an element left labeled as unimplemented in *Aset*, go to Step 4 otherwise return the solution.
4. *The optimal part*: Take an unimplemented element from *Aset*, *Aset(i)*, that has the lowest *max-level* value.
5. For each operation, *O(j)*, that implements *Aset(i),* set its cost value, *C(j)*, to its *min-adder* value as determined in the preprocessing phase and

   For each unimplemented element in *Aset*, *Aset(k)* $i \neq k$,

   a. Determine $C_{before}(k)$ by finding the *min-adder* value of *Aset(k)* when the *min-adder* values of the elements in *Aset* are assigned to 0. ($C_{before}(k)$ is the cost of implementation of *Aset(k)* at this phase of the algorithm, since all elements in *Aset* will be implemented at the end of the algorithm.)

   b. Determine $C_{after}(k)$ in the same way as done in a) but also, *assume* that the inputs of *O(j)* are in *Aset*. ($C_{after}(k)$ is the cost of implementation of *Aset(k)*, if *Aset(i)* is synthesized with *O(j)* at this phase of the algorithm.)

   c. Update the cost value, *C(j)*, as $C(j) = C(j) - (C_{before}(k) - C_{after}(k))$.
6. After the cost value of each operation, *C(j)*, is obtained, select the operation to synthesize *Aset(i)* that has the minimum cost. If there are operations that have the same minimum cost, select the operation that has the minimum *min-adder* value among these operations. Label *Aset(i)* as implemented.
7. Add the inputs of the selected operation to *Aset* that do not exist in *Aset*, label them as unimplemented, and go to Step 2.

Note that in the first iteration, the elements of *Aset* are the coefficients of the filter, and in later iterations, *Aset* includes the partial terms. Also, observe that if the algorithm returns a solution in the first iteration, the found solution is a minimum area solution.

B. *Area Optimization under a Delay Constraint: ASSUME-D Algorithm*

ASSUME-D can find a solution with either the minimum delay of the network or a user-specified delay constraint. In this paper, we describe the heuristic algorithm as it deals with the minimum delay of the network. After the network is obtained as described in [4], the *min-adder*, *min-level*, and *max-level* values of each operation and partial term are found in the preprocessing phase of the algorithm. The *min-level* is the minimum number of operations in series that implement the operation or the partial term. Consider the network of Fig. 2 with the partial term 15. The *min-level* value of 15 is 2. The minimum delay of the network, *min_delay*, is determined as the maximum of the *min-level* values of the primary outputs.

ASSUME-D synthesizes the coefficients of the filter one at a time in a top-down approach while controlling the delay. The algorithm is as follows:

1. Store the pre-processed coefficients of the filter (primary outputs of the network, all made positive and odd) in a set called *Dset* and label them as unimplemented. Assign the delay limit value of each element in *Dset* to *min_delay*.
2. Take an element labeled as unimplemented from *Dset*, *Dset(i)*, that has the highest *max-level* value. Store the operations that implement *Dset(i)* and whose *min-level* values do not exceed the *delay_limit(i)* in an empty set called *Oset*.
3. If *Dset(i)* can be implemented with an operation in *Oset* whose inputs are primary inputs or in *Dset* then, synthesize *Dset(i)* with the operation and label it as implemented. Assign the delay limit of each input of the operation, *delay_limit(j)*, to *delay_limit(i)*-1, if *delay_limit(j)* is greater than *delay_limit (i)*-1.
4. Otherwise, choose an operation from *Oset* to synthesize *Dset(i)* as done in ASSUME-A (Step 5-6) and label it as implemented. If the input(s) of the operation is not in *Dset* then, add this element to *Dset*, label it as unimplemented, and assign its delay limit value to *delay_limit(i)*-1. If the input(s) of the operation is in *Dset* then, assign the delay limit of the input, *delay_limit(j)*, to *delay_limit(i)*-1, if *delay_limit(j)* is greater than *delay_limit(i)*-1.
5. If there is an element left labeled as unimplemented in *Dset*, go to Step 2 otherwise return the solution.

IV. Experimental Results

In this section, we present the results of the proposed heuristic algorithms and compare them with the exact and heuristic algorithms. The experiments are categorized in two sets according to difficulty levels for the exact algorithms to compute a solution.

As the first experiment set, randomly generated instances between the number of 10 and 70 coefficients defined in 10 digits precision were used. There are 30 instances for each number of coefficients and the coefficients are expressed in MSD. We compared the results of proposed heuristic algorithms with the exact algorithms [10] in terms of the average additional operations. The results are given in Fig. 3.

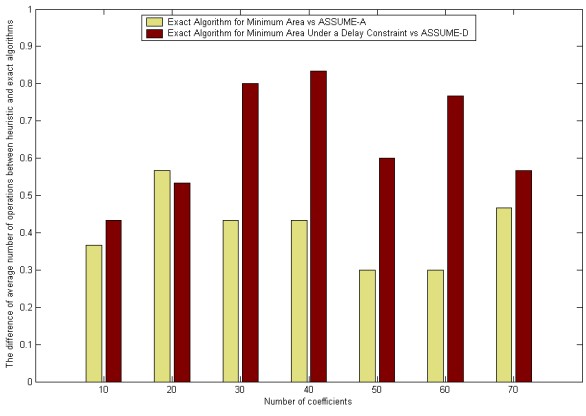

Figure 3. Results on randomly generated coefficients.

TABLE I. EXPERIMENTAL RESULTS ON FILTER INSTANCES

Filter	Minimum Area Solutions														Minimum Area under a Delay Constraint Solutions									
	CSD								MSD						CSD						MSD			
	[1]		[2]		ASSUME-A		Exact [10]		[3]		ASSUME-A		Exact [10]		[9]		ASSUME-D		Exact [10]		ASSUME-D		Exact [10]	
	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step	adder	step
1	18	3	19	3	16	3	16	3	16	3	16	3	16	3	18	3	16	3	16	3	16	3	16	3
2	25	3	25	3	23	3	23	3	23	4	22	4	22	4	26	3	23	3	23	3	22	3	22	3
3	35	3	35	3	36	3	35	3	35	3	34	3	34	3	36	3	35	3	35	3	34	3	34	3
4	57	3	55	4	50	4	49	4	51	4	48	5	47	4	57	3	50	3	49	3	48	3	47	3
5	37	3	37	4	35	4	35	4	34	4	34	4	33	4	37	3	35	3	35	3	35	3	33	3
6	58	4	55	4	52	4	51	4	50	4	49	4	*49*	*4*	58	3	52	3	52	3	49	3	*50*	*3*
7	74	4	71	4	67	4	*66*	*4*	70	4	63	4	time	time	72	3	71	3	*68*	*3*	65	3	time	time
8	101	4	96	4	90	4	*87*	*4*	91	4	85	4	time	time	95	3	93	3	*91*	*3*	86	3	time	time
Total	405	27	393	29	369	29	362	29	209	22	203	23	201	22	399	24	375	24	369	24	204	18	202	18

In this experiment, we observe that the difference of average number of operations between heuristic and exact algorithms is less than one operation, even close to half. This can be interpreted as if for half the instances we find a solution with one single extra adder and for the other half we actually find the minimum solution!

As the second experiment set, we used the filter instances presented in [3]. The experimental results are given in Table I. In this table, the first two sets of columns under CSD and MSD show the solutions for the problem of optimization of area and the last two sets of columns under CSD and MSD show the solutions for the problem of optimization of area under a delay constraint. The results are obtained on CSD and MSD representation of the coefficients, since these representations present different networks and solutions. The proposed heuristics are compared with exact [10] and heuristic [1]-[3], [9] algorithms. The results of the heuristics presented in [2] and [3] are taken from [3] and the results of the heuristics introduced in [1] and [9] are provided by the coauthor of these papers, Anup Hosangadi. In this table, *adder* denotes the number of operations and *step* denotes the maximum of number of operations in series needed to synthesize a filter. The results given in italic on the exact algorithms' column indicate that an optimal solution rather than the minimum is obtained in two hours and *time* indicates that no solution is found in two hours. The results given in the *Total* row for the MSD column are the sums of the numbers between Filter 1 and 6.

In this experiment, we observe that ASSUME-A and ASSUME-D find the same or better solutions than the other heuristics (except Filter 3 in CSD for ASSUME-A). Note that since the proposed algorithms are heuristics, ASSUME-D can find better solutions (e.g., Filter 3 in CSD) than ASSUME-A, even if ASSUME-D looks for a solution in a restricted search space. Besides, ASSUME-D can find a better solution than the optimal found by the exact algorithm (i.e., Filter 6 in MSD). We note that the average execution times of the implementations of ASSUME-A and ASSUME-D on MATLAB for the Filter 7 and 8 in MSD are 257 and 1413 seconds respectively, whereas the exact algorithms could not conclude in two hours.

V. CONCLUSIONS

In this paper, we introduced two heuristic algorithms designed for the optimization of area and optimization of area under a delay constraint. The heuristics search a solution on the network as generated in the exact algorithm and use a greedy method in choosing the operations to synthesize the coefficients. We compared our heuristics with exact and prominent heuristic algorithms proposed in this research area. It is shown that the proposed heuristics can find exact solutions, or close, on filter instances where an exact algorithm cannot conclude and find better solutions on overall instances than the existing heuristics.

ACKNOWLEDGMENT

We thank to Anup Hosangadi for his help in providing us the results of their algorithms on filter instances.

REFERENCES

[1] A. Hosangadi, F. Fallah, and R. Kastner, "Reducing hardware complexity of linear DSP systems by iteratively eliminating two term common subexpressions," Proc. of ASP-DAC, 2005.

[2] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Transactions on Circuits and Systems II, vol. 43, no. 10, pp. 677-688, 1996.

[3] I. -C. Park and H. -J. Kang, "Digital filter synthesis based on minimal signed digit representation," Proc. of DAC, pp. 468-473, 2001.

[4] P. Flores, J. Monteiro, and E. Costa, "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications," Proc. of ICCAD, pp. 13-16, 2005.

[5] A. Dempster and M. MacLeod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 42, no. 9, pp. 569-577, 1995.

[6] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," IEE Proceedings G, vol. 138, no. 3, pp. 401-412, 1991.

[7] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "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.

[8] H. -J. Kang and I. -C. Park, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," IEEE Transactions on Circuits and Systems II, vol. 48, no. 8, pp. 770-777, 2001.

[9] A. Hosangadi, F. Fallah, and R. Kastner, "Simultaneous optimization of delay and number of operations in multiplierless implementation of linear systems," Proc. of IWLS, 2005.

[10] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization of area under a delay constraint in digital filter synthesis using SAT-based integer linear programming," Proc. of DAC, pp. 669-674, 2006.

[11] E. Costa, P. Flores, and J. Monteiro, "Maximal sharing of partial terms in MCM under minimal signed digit representation.," Proc. of ECCTD, pp. 221-224, 2005.