International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

FPGA Implementation of FIR Filter Design with Optimization of Adder Tree & Constant Multiplication

I.Arivazhagan†, G.Annalakshmi†* and R.Kuppuraj†

†Department of ECE, Alpha College of Engineering & Technology, Pondicherry, India

Accepted 20 June 2015, Available online 27 June 2015, **Vol.5, No.3 (June 2015)**

Abstract

Finite Impulse Response filter is mostly used in the digital signaling processing (DSP) applications. In FIR most important parameters are complexity, cost, and power consumption. While the research focus of resolve the thus parameter to be reduced. To use the Multiple Constant Multiplication (MCM) for optimize the adder tree in the filter. In this paper we have identified the resource minimization problem in the scheduling of adder-tree operations in MCM blocks by using Mixed Integer Programming (MIP).Result shows that up to 11.09% reduction of area and 7.66% reduction of power can be achieved on the top of already optimized adder/subtractor network of the MCM block.

Keywords: *MIP, MCM, CSD, adder tree, digital signal processing, FIR etc*

1. Introduction

Digital filters are increasingly found in all areas of digital signal processing (DSP). For practical systems, it is often important that the digital filters should have low implementation cost and low power consumption, while operating at high data rates. To achieve this goal, the filter coefficients are often constrained to be integers, with single extra (possibly floating point) multiplier on the filter output. The filter requires an integer multiplier for each coefficient. These integer multipliers can be efficiently implemented using a combination of additions, subtractions and power of two shifts, collectively referred to as primitive operators. Various implementation strategies exist. The first is a binary implementation, in which each multiplier is expressed as a simple sum of power-of-two terms. The resulting design problem thus becomes that of choosing a set of filter coefficients and a corresponding implementation which offers an effective compromise between implementation complexity and performance. There are two main strategies, the first strategy is to first design an appropriate integer coefficient filter, and then to find an efficient implementation. The integer coefficient filter can be designed either by rounding the coefficients from an effective floating point design, or by employing an optimization technique such as linear programming. The optimal SPT representation is easily found using the canonic signed digit (CSD) algorithm. The optimal graph design is a more difficult problem, although various heuristic methods exist, which give a close to optimal implementation. The problem with this approach is that it achieves compromise which tends to favor higher performance and higher complexity solutions. The reason for this is due to the choice of filter being made without reference to its complexity.

An alternative method is to use a complexity constrained approach, where a specific implementation is selected with an appropriate complexity constraint. The space of possible parameters is then searched in order to find a filter with optimum performance given the constraints. Although this approach has been found to work well with an SPT implementation, it is not well suited for use with directed graphs, due to the epistatic nature of the problem. A small change in the graph will typically lead to a large change in filter characteristics. This makes it very difficult to find an optimal solution. Multiplication with a constant fixed-point number is a basic operation in many digital signal processing applications. A multiplication can be implemented using a network of binary shifts and adders (or subtractors). For bit-parallel arithmetic the shifts (multiplication by a power of two) can be hardwired and therefore do not require any gates. The hardware cost can thus be approximated with the required number of adders and subtractors. For two's complement numbers, the hardware cost of a subtraction is approximately equal to that of an adder.

For convenience both adders and subtractors will therefore be referred to as adders. Since a fixed-point number can be converted to an integer number by multiplication by 2^N for some N, only integer numbers.

*Corresponding author: **G.Annalakshmi**

By using carry-save adders the need for carry propagation in the adder is avoided and the latency of one addition is equal to the gate delay of a full adder. The carry-save adder has three inputs and two outputs, where the two outputs together form the result.

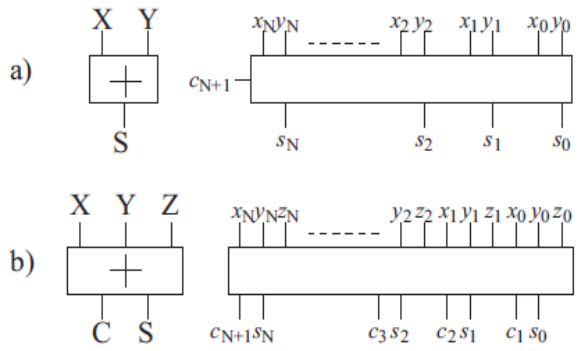

Fig.1 a) Carry-propagation adder symbol and structure. b) Carry-save adder symbol and structure

A variable can be multiplied by a given set of fixed-point constants using a multiplier block that consists exclusively of additions, subtractions, and shifts. The generation of a multiplier block from the set of constants is known as the multiple constant multiplication (MCM) problems. Finding the optimal solution, i.e., the one with the fewest number of additions and subtractions is known to be NP-complete. We propose a new heuristic algorithm for the MCM problem, which finds solutions that require up to 20% less additions and subtractions than the solutions found by the best previously known algorithm. At the same time, our algorithm is not limited by the constant bitwidths, in contrast to the closest competing algorithm. We present our algorithm using a unifying formal framework for the best, graph-based MCM algorithms and provide a detailed runtime analysis and experimental evaluation. We show that our algorithm can handle problem sizes as large as 100 32-bit constants in a time acceptable for most applications.

2. Existing System

2.1Digital serial filter

A limiting factor in many modern DSP systems is the power consumption. This is due to two different problems. First, when the systems becomes larger and whole systems are integrated on a single chip, i.e., System-on-Chip (SoC), and the clock frequency is increased, the total power dissipation is approaching the limit when an expensive cooling system is required to avoid overheated chips. Second, the portable equipment such as cellular phones and portable computers are becoming increasingly popular. These products use batteries as their power supply. A decrease in power consumption increases the portability since smaller batteries can be used with longer life-time between recharges. Hence, design for low power consumption is important. In this thesis, we will assume working with a hard real-time system, i.e., all operations should be performed within the sample period. Hence, throughput or the sample rate is not a cost function it is a requirement. The major cost function becomes low power consumption. In older CMOS processes the area was a limiting factor due to high manufacturing costs. In modern deep submicron CMOS processes the area is no longer a problem. However, the close relationship between area and switching capacitance makes it still interesting to reduce chip area in order to reduce the power consumption. The design time is also a key factor for low price products. Hence, an efficient design process aiming at first time right silicon is of a great interest for minimizing total costs.

Due to the intensive use of FIR filters in video and communication systems, high performance in speed, area and power consumption is demanded. Basically, digital filters are used to modify the characteristic of signals in time and frequency domain and have been recognized as primary digital signal processing. In DSP, the design methods were mainly focused in multiplier-based architectures to implement the multiply-and-Accumulate (MAC) blocks that constitute the central piece in FIR filters and several functions. The FIR digital filter is presented as

$$y[n] = \sum_{k=0}^{N-1} c[k]x[n-k] \quad (1)$$

Where $y[n]$ is the FIR filter output, $x[n-k]$ is input data and $c[k]$ represents the filter coefficients Equation shows that multiplier-based filter implementations may become highly expensive in terms of area and speed. This issue has been partially solved with the new generation of low-cost FPGAs that have embedded DSP blocks. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches.

In literature, several multiplier less schemes had been proposed. These methods can be classified in two categories according to how they manipulate the filter coefficients for the multiply operation. The first type of multiplier-less technique is the conversion-based approach, in which the coefficients are transformed to other numeric representations whose hardware implementation or manipulation is more efficient than the traditional binary representation. Example of such techniques are the Canonic Sign Digit (CSD) method, in which coefficients are represented by a combination of powers of two in such a way that multiplication can be simply implemented with adder/subtractors and shifters, and the Dempster-Mcleod method, which similarly involves the representation of filter coefficients with powers of two but in this case arranging partial results in cascade to introduce further savings in the usage of adders.

The second type of multiplier-less method involves use of memories (RAMs,ROMs) or Look-Up Tables (LUTs) to store pre-computed values of coefficient operations. These memory-based methods involve Constant Coefficient Multiplier method and the very-well known Distributed Arithmetic method as examples. Distributed Arithmetic (DA) algorithm appeared as a very efficient solution especially suited for LUT-based FPGA architectures. Croisier et al had proposed the multiplier less architecture of DA algorithm and it is based on an efficient partition of the function in partial terms using 2's complement binary representation of data. The partial terms can be pre-computed and stored in LUTs. Yoo et al. observed that the requirement of memory/LUT capacity increases exponentially with the order of the filter, given that DA implementations need 2K – words, *K* being the number of taps of the filter.

The work in this paper presents the design and implementation of serial and parallel distributed algorithm for FIR filter target as Spartan 3E FPGA. The results of the implementation experiment are analyzed in terms of parameters such as area and speed. The brief description of the distributed algorithm is presented. The implementation of the proposed technique for FIR filters is discussed. The Section 4 presents the implementation results. The last section concludes the work and presents the future work.

2.2 Gate- level optimization

Reducing power consumption is both an imperative and a daunting challenge for today's ASIC and IC designers. Silicon technology advances have made it possible to pack hundreds of thousands of transistors on a single chip. In addition, performance goals require high clock speeds posing difficult power dissipation and distribution problems. Further complicating the mix is the rapidly increasing demand for power-sensitive applications like high-performance computer systems; portable, battery-operated computers, medical devices and telecommunications equipment.

In response, designers are moving to incorporate power considerations into all phases of their design flow. In methodologies that dictate the use of logic synthesis tools on some portion of the design, such tools must also consider power consumption when performing design trade-offs. Until recently, commercial synthesis tools have mainly focused on timing and area optimization. Power reduction of a circuit, however, can come at the expense of these design goals, hence must operate within these design constraints and perform tradeoffs between them.

2.2.1 Power cost function

Power compiler optimizes a design according to a set of constraints that are defined by the user and the technology library. These constraints are used to determine the cost of a given design which guides the optimization algorithms. The cost measures the extent to which a constraint has been met. If a constraint has been satisfied, the corresponding cost will be zero. The application of optimization algorithms is divided into two phases. The two phases are distinguished by the cost functions being optimized. The first phase uses an optimization only cost function, while the second phase adds a design rule cost. The optimization cost function is shown below in order of importance. Some components might not be active on a given design

1. Maximum
2. Minimum delay
3. Maximum dynamic power
4. Maximum leakage power
5. Maximum area

A prioritized cost function means that timing constraints will not be save power, but available timing slack can be consumed if power can be reduced. In the design rule phase, the cost function is identical to the optimization cost function expect that a design rule cost is added as the important cost. Design rule constraints reflect technology-specific restrictions that must be met for a functional design, such as the maximum signal transition time for nets. Cost function components are evaluated independently in order of importance. A transformation is accepted if it decreases the cost of one component without increasing more important costs. Optimization stops when all costs are zero, or no further improvement can be made to the cost function. Cell internal power updates occur not only when a cell is added , deleted, or sized but also when input transition times or the output load of a cell changes. The latter situations can be easily detected such that internal power updated only occur on cells that have changed their transitions times or capacitance.

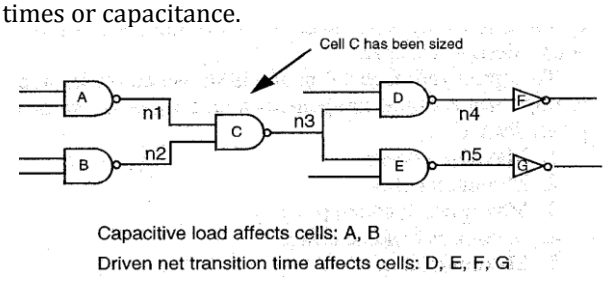

Fig. 2 Incremental internal power updates

For switching power, activities of only the new or changed nets are recomputed by the probabilistic simulation engine when the connectivity or local functionality of a design changes. The switching activities of the unchanged nets serve as start points for propagation. This method is referred to as local propagation and is very fast since the BDD needed to calculate the switching activity is small.

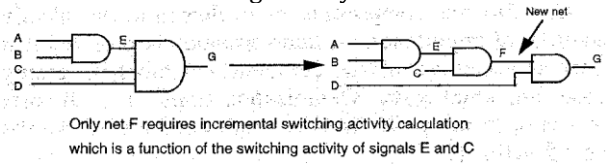

Fig.3 Local propagation

Since the BDDs are not built with respect to the primary inputs, some accuracy is lost due to less spatial correlation information. However, absolute accuracy loss is acceptable as long as the relative accuracy of the switching activities between nets is maintained.

3. Proposed System

3.1 Adder Tree Scheduling

3.1.1 Greedy Adder-Tree Scheduling

The common practice of handling the summation of CS terms of each coefficient is to use the tree-height minimization algorithm to produce a height optimum adder-tree. The Tree-height minimization algorithm iteratively collapses the pair $\{T_i,T_j\}$ with smallest delays using an ADD/SUB to form a new term with delay $\max(D_i, D_j) + 1$, until a single term is reduced to. Fig. 4 gives an example of the schedule for an adder-tree on the left with minimum delay. Note that either a positive or negative sign is associated with each input term (see Fig. 4(a)), which denotes whether the corresponding term should be added to or subtracted from the summation. These signs also determine whether an addition operation or a subtraction operation should be used when the algorithm collapses a pair of terms in the adder-tree based on the following rules.

(1) If two input edges are of the same sign, an ADD will be used; otherwise, it will be a SUB.

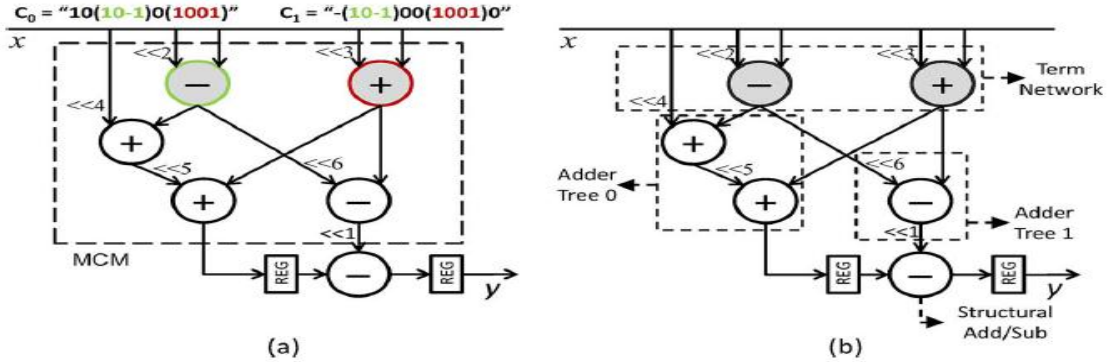

Fig.4 Composition of the MCM block (a) MCM and common sub expressions. (b) Term network and adder-trees for each coefficient

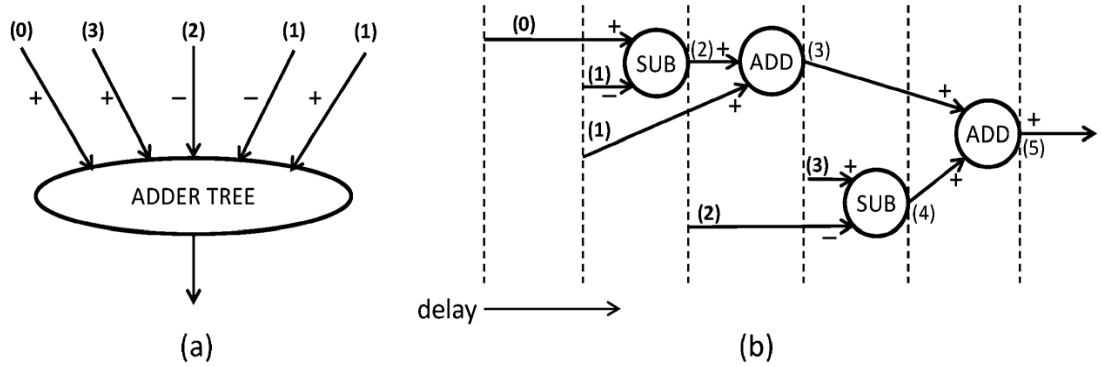

Fig. 5 (a) An example adder-tree with delays and signs on each input term, (b) An internal schedule with minimum delay

(2) The sign of the output edge is always the same as that of the left input edge (i.e., the minuend edge in the subtraction case). Using these two rules, it is possible that the final term producing the summation result may carry a negative sign, such that a negation is needed after the adder-tree to correct the value. For an FIR filter [1], results from multiple adder-trees are accumulated by a structural adder-register line. So the negation can be eliminated by replacing the structural adder with a subtractor (see coefficient C_i in Fig. 4 for an example).

3.1.2 Cost Model

In order to quantify and minimize the hardware cost of the adder-tree, we model the cost of ADD/SUB operations in this section based on the ripple carry implementation, which is most area efficient and will be picked up by the hardware compiler whenever the timing allows. Without loss of generality, for a single ADD/SUB operation, the pair of its input operands may be of different bit-widths, and one of them is to be left shifted by certain bit positions. We enumerate all the

scenarios of shift-add/sub operations in Fig. 5. The cost calculation is done separately in three bit-segments. Starting from the least significant bit (LSB), the 1st segment covers the bit positions up to but not including the first bit of the shifted operand; the 3rd segment covers the bits corresponding to the sign Cases or SUB. Extension bits of the sign extended operand; the 2nd segment takes the rest of the bit positions.

Two cases of ADD operation are shown in Fig. 6 (a) and (b). In both cases, the 2nd and 3rd segments are implemented by one Full Adder (FA) per bit, while the 1st segment cost nothing than wiring. Four cases of SUB operation are shown in Fig. 6(c)-(f). In the first two cases where the shift is with the minuend, the 1st segment is implemented by FAs with invertors on the minuend bits, except for a single special case at the LSB using direct wire connection1 to save a pair of FA and invertors.

The 1 st segments for the last two cases are wires. The 2nd segment for all cases is implemented in pairs of FAs and invertors. For the 3rd segment, when the sign extension bits are from the subtrahend, inverters are not needed since these bits simply take the value of the inverted sign of the subtrahend. This cost model is verified experimentally from synthesis results of Synopsis Design Compiler for ASICs, and is applicable to cases where either or both of input operands are unsigned signals.

3.1.4 Logic Depth Relaxation

The clock performance of the entire FIR filter is decided by the largest of the delays of all coefficients. Assuming the delay of an ADD/SUB operator to be 1 unit, the delay of the constant multiplication by a coefficient can be simply measured by the number of ADD/SUB steps on a maximal path in the part of the network corresponding to the coefficient.

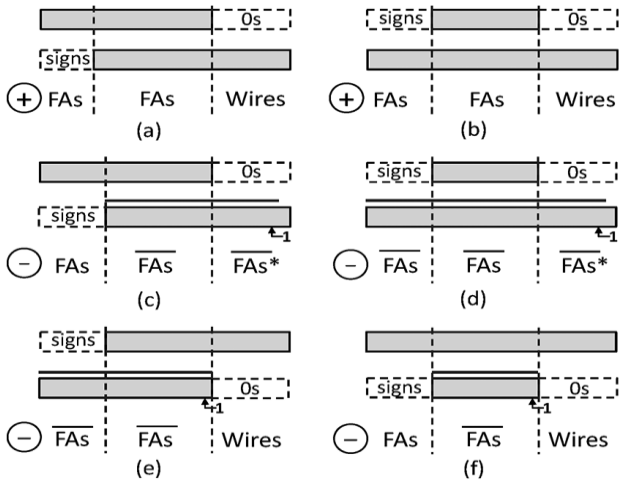

Fig. 6 Cost of ADD/SUB operation under various input scenarios. Notations: FA—Full Adder, above line—invertor (INV). (a)(b) Cases for ADD. (c)(d)(e)(f)

We generally use logic depth to describe the required ADD/SUB steps. For a coefficient whose logic depth is less than the filter's logic depth, incrementing (relaxing) its logic depth may reduce the resource consumption. Given an algorithm which computes the adder-tree of the minimum resource on a given depth L for a coefficient, if L is less than the filter's logic depth, one can always try increasing L by 1 and rescheduling onto a L + 1 depth adder-tree for possible reduction of resource without degrading the filter's clock performance.

3.2 MCM

Critical path and shortest path solving contribute to most of the computation time in retiming.

Definition 1 (the path solver problem)

Let, S= $\{s_0,s_1,s_3,\ldots,s_k\}$ where is the maximum number of feasible solutions available for retiming of a considered filter DFG. During retiming of digital filters in high level synthesis, the shortest path between the nodes must be computed for (K+1) times where is the number of feasible solutions available for the DFG which is nothing but unique entries in path delay matrix. Similarly, the critical path must be computed for (K+1) . General purpose processors (GPPs) where retiming algorithm is implemented are fully programmable but are less efficient in terms of power and performance. Hence, the problem is to improve the performance and power of retiming using FPGA based path solvers. Further, along with retiming, high level transformation technique called automatic pipeline is applied to improve the filter speed.

Definition 2 (multiple constant multiplication in digital filters)

For the considered filter coefficient constant in the retimed filters, find the set of multiplier less operations $(O_1,O_2,O_3,\ldots O_n)$ with minimum number of addition, subtraction, and shift operations using multiple constant multiplier architecture to optimize the filter architecture further.

Definition 3 (optimization and automation of filter HDL)

An environment needs to be developed to obtain HDLs of retimed filters in which user can choose different data path element architectures depending on the specifications. This reduces time to market and helps to evaluate a lot of hardware implementation trade-offs. Filter equivalence checking after applying high level transformation needs to be done which needs to be developed as a part of the optimization environment.

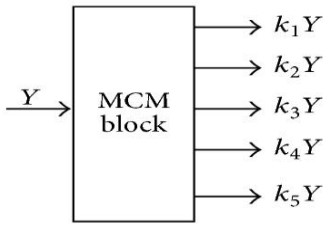

Fig.7 Example for addressing MCM problem in digital filters

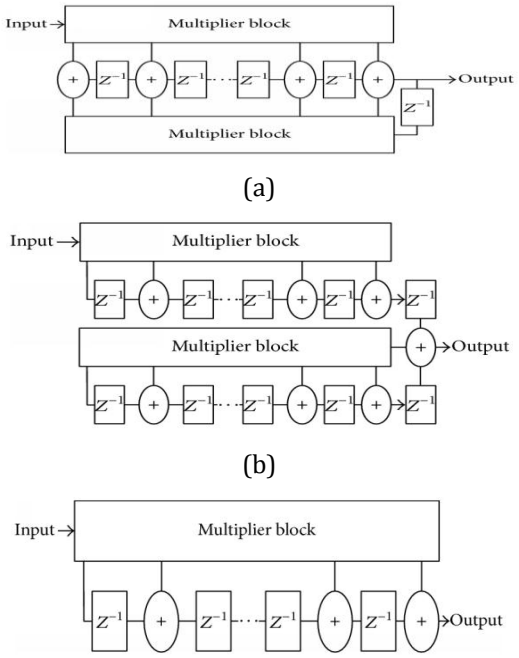

Fig.8 General structure of MCM block for (a) FIR filter, (b) transposed direct form-I IIR filter, and (c) transposed direct form-II IIR filter.

3.2.1 Principle of Shortest Path and MCM Algorithm

Several FPGA synthesis algorithms have been proposed specifically for sequential circuits. However, in this paper, authors suggest a method for efficient retiming process using FPGA based path solvers. This can be applied to any retiming techniques available in literature. Shortest path is solved in filter DFG using Floyd-Warshall algorithm. The Floyd-Warshall algorithm uses an approach of dynamic programming to solve the shortest-paths problem on a DFG. The Floyd-Warshall Algorithm can solve the shortest path problem in $O(n^3)$ time where is the number of nodes in the DFG. Let denote the weight of the shortest path from i and j to such that all intermediate vertices are contained in the set {1,2,…,} . That is, the path is decomposed into i →K → j. Let the vertices in the graph be numbered from1, 2,…, n . Consider the subset {1, 2,,…, K}of these n vertices. Find the shortest path from vertex to vertex that uses vertices in the set {1,2,…,K}only. Then, there are two situations possible

k is an intermediate vertex on the shortest path

i. k is not an intermediate vertex on the shortest path

If the vertex is not an intermediate vertex on P, then

$$d_{ij}(k) = d_{ij}(k-1)\ else\ d_{ij}(k) = d_{ik}(k-1) + d_{kj}(k-1) \quad (2)$$

In either case, the sub paths contain nodes from {1,2,…,(K-1)}. Therefore,

$$d_{ij}(k) = d_{ij}(k-1) + d_{kj}(k-1) \quad (3)$$

When K=0, then

$$d_{ij}(0) = \{W_{ij}\}, \quad (4)$$

And if k0 then

$$d_{ij}(k) = \min\{d_{ij}(k-1) + d_{ij}(k-1)\} \quad (5)$$

Let D be the incidence matrix with the graph edge weight information W initially. D is then updated with the calculated shortest paths;

(1) n=≠of rows in W, D^0=w
(2) for (k=1 to n)
(3) for (i=1 to n)
(4) for(j=1 to n)
(5) $d_{ij}(k) = \min\{d_{ij}(k-1), d_{ik}(k-1) + d_{ik}(k-1)\}$
(6) end for
(7) end for
(8) end for
(9) return D^n

Algorithm 1

The final D matrix will store all the shortest paths. This algorithm is extended for retiming of digital filters. The multiple constant multiplication (MCM) problem is addressed in the literature using either graph based methods or using common sub expression elimination method. In common sub expression elimination algorithm, all possible sub expressions are extracted for a variable. But this is possible only if it is defined as minimum signed digit and as canonical signed digit. Then the sub expression is found such that it can be shared by multiple constant multiplication values. In this paper, the above two concepts are extended for automatic pipelining and retiming of digital filters in high level synthesis. In all the digital filters, the filter coefficients are known beforehand. Hence, full flexibility of the multiplier is not necessary and we can

make use of MCM designs. This method is more efficient when compared to shift and add multiplications as intermediate results can be shared which reduces the area of multiplier less implementation of digital filters. The sharing of intermediate result will provide potential area saving with increased filter order.

Consider the filter coefficient set which is to be used for the filter design given by T={c_1,c_2,c_3,...c_n} , we need to find the smallest set S given by {a_1,a_2,a_3,....,s_1,s_2,s_3,...,} Where a (adders/Subtractors) & s (shifts)<S such that the set is made of adder/subtracters, shifters, and operations. Here, shift operations also can be shared across multiple points so that the output set is optimum. Here algorithm is used to generate corresponding DFG for the multiplier block implementing the parallel multiplications C_1 *X,C_2.*X,.....,C_n .The only operations used in the generated DAG and input design matrices are additions, subtractions, shifts, and negations. In this paper, performance of MCM based filter designs is further improved by combining this approach with retiming. The multiplier less filter circuit is further retimed to reduce the overall clock period which increases the clock frequency.

Consider $l_1$and l_2 as two integers which specifies left shifts and specifies right shift and r ≥ 0 let be the sign bit which can be {0,1}.An A operation is an operation with two integer inputs u and v and one fundamental output which is defined as

$$A_p(u,v) = |(u \ll l_1) + (1)s(v \ll l_2)| \gg r = 2^{l_1}u + (-1)^s 2^{l_2}v|2^{-r} \quad (6)$$

Where «is a left binary shift, » is a right binary shift, and P = {l_1,l_2,r,s} is the parameter set or the A configuration of A_p.To preserve all significant bits of the output,2^r must divide2^{l2}u+(-1)^s2^{l_2}v.The left shifts are limited to the bit width of the target. All A operations are used to build A-graph, for a given set of target filter set of target filter coefficients C, We can find set S such that multiplier less digital filter is designed.

3.3 Mixed integer programming

A mixed-integer programming (MIP) results when some of the variables in your in your model are real-valued (can take on fractional values) and some of the variables are integer-valued. The model is therefore mixed. When the objective function and constraints are all linear in form, then it is a mixed-integer linear program (MILP). In common parlance, MIP is often taken to mean MILP, though mixed-integer nonlinear program (MINLP) also occur, and are much harder to solve. MILP techniques are effective not only for mixed problems, but also for pure-integer problems, pure-binary problems, or in fact any combination of real-, integer-, and binary-valued variables.

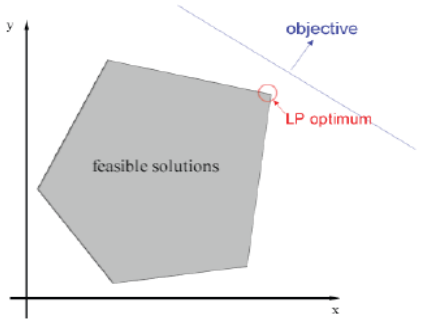

Fig.9 Mixed integer programming

Mixed-integer programs often arise in the context of what would otherwise seem to be a linear program. However, as we saw in the previous chapter, it simply doesn't work to treat the integer variable as real, solve the LP, then round the integer variable to the nearest integer value.The problems most commonly solved by the Gurobi Parallel Mixed Integer Programming solver are of the form

Objective: minimize c x

Constraints: A x = b (linear constraints)
l ≤ x ≤ u (bound constraints)
some or all x_j must take integer values (integrality constraints)

The integrality constraints allow MIP models to capture the discrete nature of some decisions. For example, a variable whose values are restricted to 0 or 1, called a binary variable, can be used to decide whether or not some action is taken, such as building a warehouse or purchasing a new machine.

The GurobiMIP [1] solver can also solve models with a quadratic objective and/or quadratic constraints:

Objective: minimize x Q x + q x

Constraints: A x = b (linear constraints)
l ≤ x ≤ u (bound constraints)
x Q x + q x ≤ b (quadratic constraints)
some or all x must take integer values (integrality constraints)

MIP models with a quadratic objective but without quadratic constraints are called Mixed Integer Quadratic Programming (MIQP) problems. MIP models with quadratic constraints are called Mixed Integer Quadratically Constrained Programming (MIQCP) problems. Models without any quadratic features are often referred to as Mixed Integer Linear Programming (MILP) problems. What follows is a description of the algorithm used by Gurobi to solve MILP models. The extension to MIQP and MIQCP is mostly straightforward, but we won't describe them here.

3.3.1 Branch-and-Bound

Mixed Integer Linear Programming problems are generally solved using a linear programming based

branch and bound algorithm. Basic LP based branch and bound can be described as follows. We begin with the original MIP. Not knowing how to solve this problem directly, we remove all of the integrality restrictions. The resulting LP is called the linear programming relaxation of the original MIP. We can then solve this LP. If the result happens to satisfy all of the integrality restrictions, even though these were not explicitly imposed, then we have been quite lucky. This solution is an optimal solution of the original MIP, and we can stop. If not, as is usually the case, then the normal procedure is to pick some variable that is restricted to be integer, but whose value in the LP relaxation is fractional. For the sake of argument, suppose that this variable is x and its value in the LP relaxation is 5.7. We can then exclude this value by, in turn, imposing the restrictions $x \leq 5.0$ and $x \geq 6.0$.

If the original MIP is denoted P_0, then we might denote these two new MIPs by P_1, where $x \leq 5.0$ is imposed, and P_2, where $x \geq 6.0$ is imposed. The variable x is then called a branching variable, and we are said to have branched on x, producing the two sub-MIPs P_1 and P_2. It should be clear that if we can compute optimal solutions for each of P_1 and P_2, then we can take the better of these two solutions and it will be optimal to the original problem, P_0. In this way we have replaced P by two simpler (or at least more-restricted) MIPs. We now apply the same idea to these two MIPs, solving the corresponding LP relaxations and, if necessary, selecting branching variables. In so doing we generate what is called a search tree. The MIPs generated by the search procedure are called the nodes of the tree, with P_0 designated as the root node. The leaves of the tree are all the nodes from which we have not yet branched. In general, if we reach a point at which we can solve or otherwise dispose of all leaf nodes, then we will have solved the original MIP.

3.3.1.1 Fathomed and Incumbent Nodes

To complete our description of (LP based) branch and bound we need to describe the additional logic that is applied in processing the nodes of the search tree. Let us assume that our goal is to minimize the objective, and suppose that we have just solved the LP relaxation of some node in the search tree.

Fig.10 Each node in branch and bound is a new MIP

If it happens that all of the integrality restrictions in the original MIP are satisfied in the solution at this node, then we know we have found a feasible solution to the original MIP. There are two important steps that we then take. First, we designate this node as fathomed. It is not necessary to branch on this node; It is a permanent leaf of the search tree. Second, we analyze the information provided by the feasible solution we have just found, as follows. Let us denote the best integer solution found at any point in the search as the incumbent. At the start of the search, we have no incumbent. If the integer feasible solution that we have just found has a better objective function value than the current incumbent (or if we have no incumbent), then we record this solution as the new incumbent, along with its objective function value. Otherwise, no incumbent update is necessary and we simply proceed with the search.

There are two other possibilities that can lead to a node being fathomed. First, it can happen that the branch that led to the current node added a restriction that made the LP relaxation infeasible. Obviously if this node contains no feasible solution to the LP relaxation, then it contains no integer feasible solution. The second possibility is that an optimal relaxation solution is found, but its objective value is bigger than that of the current incumbent. Clearly this node cannot yield a better integral solution and again can be fathomed.

3.3.1.2 Best Bound and Gap

There are two additional important values we need to introduce to complete our description of branch and bound. First observe that, once we have an incumbent, the objective value for this incumbent, assuming the original MIP is a minimization problem, is a valid upper bound on the optimal solution of the given MIP. That is, we know that we will never have to accept an integer solution of value higher than this value. Somewhat less obvious is that, at any time during the branch and bound search we also have a valid lower bound, sometimes call the best bound. This bound is obtained by taking the minimum of the optimal objective values of all of the current leaf nodes. Finally, the difference between the current upper and lower bounds is known as the gap. When the gap is zero we have demonstrated optimality.

4. Experimental Result

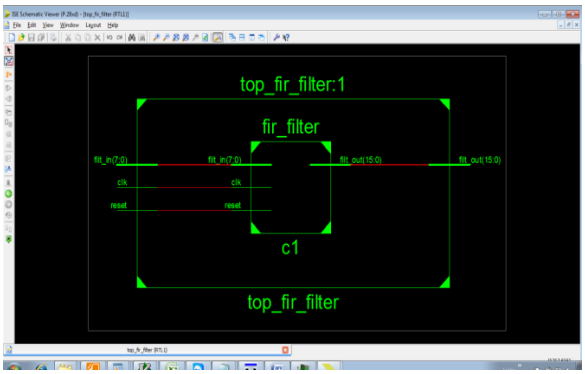

Fig.11 Schematic View of FIR Filter

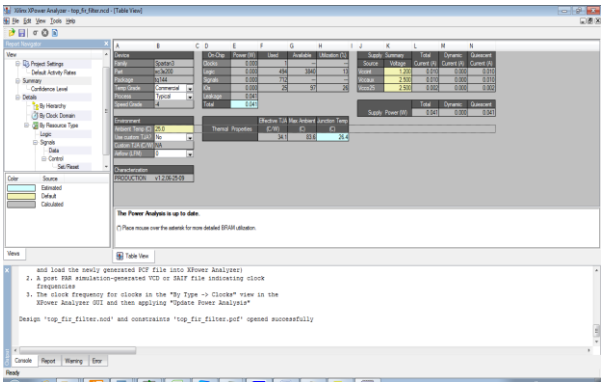

Fig.12 Power Report

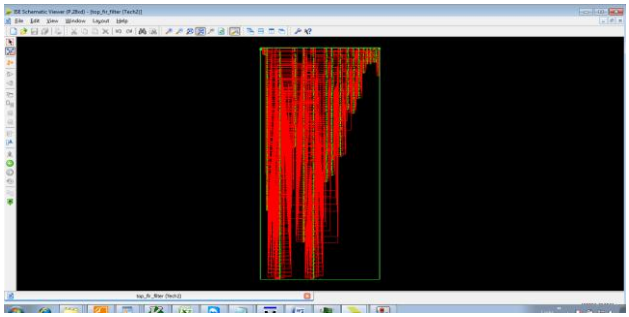

Fig.13 Full RTL Schematic

FIR filter layout is designed in the xilinix software. It describes the input (8-bit), clock, reset and output (16-bit). We have designed this filter to get 16-bit output from 8-bit input. Additionally clock is used to synchronize the signal. The reset pin is used to reset the FIR filter. We have used this technique in four types of filters (LPF, HPF, BPF, and BSF). The inputs are got from the NCO. The layout is created by using the code and link from the ModelSim software.

The power report describes about the temperature and the power consumption of each sections like clock, signal, logic function, input output ports and the important section of power leakage. Thus, the parameters are reduced as much as compared to the existing system.The RTL schematic view represents and describes about layout of the chip, arrangement and connections between the gates. It is used to study about the structure of the filter in the integrated chip.

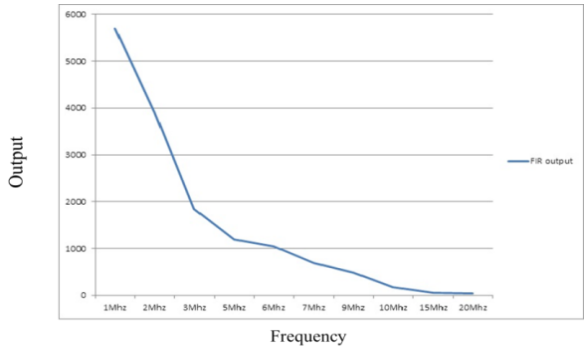

Fig.14 Frequency Response for LPF

Table.1. Comparison of Existing and Proposed system

	Filter Order	Cell Area (sq. um)			Power Consumption (mW)		
		Conv.	MIP	Impro.%	Conv.	MIP	Impro.%.
Existing	64	57026	52925	7.19	1.064	1.013	4.80
Proposed	64	57026	51701	9.337	1.064	0.9912	6.842
	64	57026	50701	11.091	1.064	0.9234	7.665

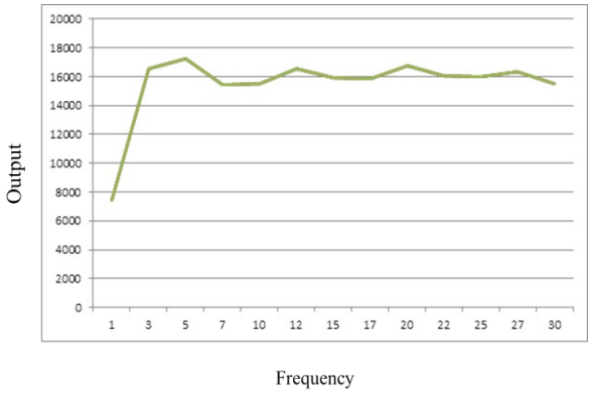

Fig.15 Frequency Response for HPF

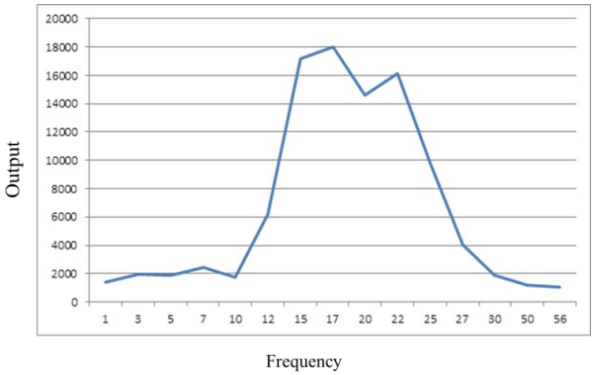

Fig.16 Frequency Response for BPF

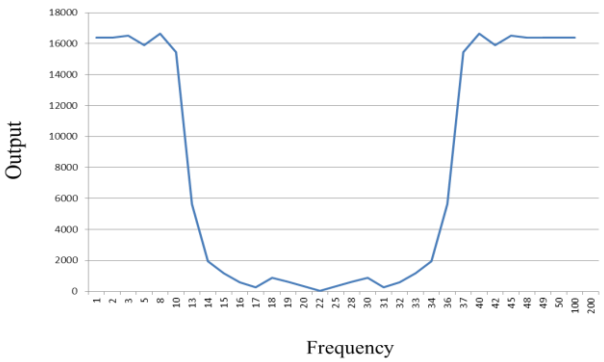

Fig.17 Frequency Response for BSF

The frequency response diagram implies the frequency characteristics of filters like low pass filter, high pass

filter, band pass filter and band stop filter from the ModelSim software.

4.5 Comparison

The below tabulations imply the cell area and the power consumption for the existing and the proposed system. The cell area column has three parts. They are conventional type to cover maximum area, MIP to reduce the usage of the area as much as possible by using respective algorithm to use and Improvement column defines, how much the area is reduced. And the power consumption is as same as the cell area of the gate.

Conclusion

We have identified the resource minimization problem in the scheduling of adder-tree operations for the MCM block of transposed direct-form FIR filter, and presented an MIP-based algorithm for exact bit-level resource optimization. Result shows that up to 11.09% reduction of area and 7.66% reduction of power can be achieved on top of already optimized ADD/SUB networks of MCM blocks. Further exploration of efficient heuristic algorithms for resource minimization of adder-trees of FIR filters could be done in the future.

References

Yu Pan and Pramod Kumar Meher (February 2014), Bit-Level Optimization of Adder-Trees for Multiple Constant Multiplications for Efficient FIR Filter Implementation Senior Member, IEEE circuits and systems—i: regular papers, vol. 61, no. 2

D.R.Bulland D. H. Horrocks (Jun. 1991), Primitive operator digital filter, IEE Proceedings-G,vol. 138, no. 3, pp. 401-412

A.G.Dempsterand M. D. Macleod (1995), Use of minimum-adder multiplier blocks in FIR digital filters, IEEE Trans. Circuits Syst. II, Analod Digit. Signal Process., vol. 42, no. 9, pp. 569-577.

S. D. S. M. Mehendale and G. Venkatesh (1995), Synthesis of multiplier-less FIR filters with minimum number of additions, in Proc. IEEE ICCAD.

I.C. Parkand H. J. Kang (2001), Digital filter synthesis based on minimal signed digit representation, in Proc. Design Autom. Conf. (DAC).

Y. Voronenko and M. Puschel (2007), Multiplierless multiple constant multiplication, ACM Trans. Algorithms,vol. 3,no. 2.

P. K. Meher and Y. Pan (Oct. 2011), MCM-based implementation of block fir filters for high-speed and low-power applications, in Proc. VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th Int. Conf., pp. 118-121.

L. Aksoy, C. Lazzari, E. Costa, P. Flores, and J. Monteiro (Mar. 2013), Design of digit-serial FIR filters: Algorithms, architectures, and a CAD tool, IEEE Trans. Very Large Scale Integration (VLSI) Syst., vol. 21, no. 3, pp. 498-511.

M. B. Gately, M. B. Yeary, and C. Y. Tang (May 2012), Multiple real-constant multiplication with improved cost model and greedy and optimal searches, in Proc. IEEE ISCAS, pp. 588-591.

M. Kumm, P. Zipf, M. Faust, and C.-H. Chang (May 2012), Pipelined adder graph optimization for high speed multiple constant multiplication, in Proc. IEEEISCAS, pp. 49-52.

R. Hartley and A. Casavant (Nov. 1989), Tree-height minimization in pipelined architectures, in Proc. IEEE ICCAD.

R. Mahesh and A. Vinod (2008), A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters, IEEE Trans. Computer-Aided Des. Integr. Circuits Syst., vol. 27, no. 2, pp. 217-229.