International Journal of LNCT, Vol 2(5)
ISSN (Online): 2456-9895

An comprehensive study weighted partitioning for multiple-constant convolution circuit to design fast multipliers

Deepak Tiwari[1] and Prashant Purohit[2]
M.Tech Scholar, LNCT Bhopal, India[1]
Assistant Professor, LNCT Bhopal, India[2]

Abstract
The graph partitioning issue are utilized as a part of performance radar systems. The circle channel is connected in a criticism circle. The circle channel is a FIR channel. FIR channels can be actualized with multiplier hinders using Multiple Constant Multiplication(MCM). MCM replaces consistent augmentations by adders and shifts. Movements can be hardwired in usage so the last cost of the plan can be estimated by the quantity of adders/subtractors. For longer sift where consistent increase turns through more cumbersome MCM makes such plans basic. There are a few algorithms to take care of the MCM issue. In this work method for quick convolution algorithms is to express convolution activities with the ideas and apparatuses of another numerical field. This encourages it to apply techniques from this field to the first issue is a foundation of transformation based fast convolution algorithms-like FFT-based fast convolution. It makes utilization of polynomial insertion to determine streamlined conditions for discrete convolutions, with less terms than the discrete convolution entirety. To wrap things up numbered hypothesis prompt many, regularly progressive, new methodologies. Huge numbers of these systems were considered in the zone of quick Fourier change algorithms and after that later connected to convolution techniques too. This work exhibits a broad review on numerous consistent convolution quick multiplierless Circuit in view of weighted partitiioning algorithm.

Keywords
Constant convolution, Multiplierless circuit, Weighted partitioning.

1.Introduction
Multiplication is a fundamental operation performed in the convolution. The way a multiplication is carried out in ASIC and FPGA designs initially seems to be very similar. Both ASICs and FPGAs require the same algorithms to be implemented. For example the structure of parallel-array multipliers [Was78] or Wallace tree multipliers [Wal64] for FPGAs and ASICs are very similar. Nevertheless, the most important advantage of FPGAs over ASICs is reconfiguration which allows for a change of the multiplication coefficient either by the change of the multiplicand (an input to a fully functional multiplier denoted further as Variable Coefficient Multiplier VCM) or by the change of a Constant Coefficient Multiplier (KCM) circuit. The KCM, in comparison to the VCM, has much lower hardware requirements [Cha96, Pet95, Wia01a], and therefore is recommended providing that a coefficient value is relatively constant during the calculation process.
A constant coefficient multiplier is usually implemented in a multiplierless fashion by using only shifts and adders from the binary representation (BR) of the multiplicand. For example, A multiplied by B= 14= 11102 can be implemented as (A<<1)+(A<<2)+(A<<3), where '<<' denotes a shift to the left. It should be noted that the hardware requirements depend on the choice of a coefficient, i.e. the number of 1's in the binary representation of the coefficient should be as low as possible. Therefore several algorithms have been developed in order to reduce hardware by a proper choice of the multiplication coefficient (e.g. for FIR filters design. However, in this paper the assumption is made that the value of the coefficient is an input parameter to the design and therefore the coefficient value cannot be changed.

An alternative to the MAC approach is DA which is a well known method to save resources and was developed in the late 1960's independently by Croiser et al. and Zohar. The term "distributed arithmetic" is derived from the fact that the arithmetic operations are not easily apparent and often distributed across the terms. This can be verified by looking which is a rearranged. DA is a bit-level rearrangement of constant multiplication, which replaces multiplication with a high number of lookup tables and a scaling accumulator. Using a DA method, the filter can be implemented either in bit serial or fully parallel mode to tradeoff between bandwidth and area utilization. In essence, this replicates the lookup tables, allowing for parallel lookups. Therefore the multiplication of multiple bits is performed at the same time.

2.Graph partitioning
Graph partitioning, also called k-way partitioning, denotes the problem of putting each vertex of a graph into one of k blocks in a way that maximizes the

quality of the partition. minimizing the dependencies/communication between nodes. The corresponding decision problem to graph partitioning is NP-complete [14] and since the graphs worth distributing are big, heuristics are used in practice. It has also been shown that even finding approximations of guaranteed quality is NP-hard in some cases, for greater detail refer to [9, 14, 5]. There is a huge amount of work done on graph partitioning which led to an enormous diversity of heuristics used, from branch-and-bound over flow algorithms that are based on computing maximum flows of a graph to spectral partitioning which performs computations on the adjacency matrix of the input graph.

The idea of Graph partitioning can expedite the process of querying huge graphs. But, the overhead with Graph partitioning is that may lose some of the results due to the cut edge. Any edge that is cut in order to partition a graph into smaller graphs can be called as cut-edge. It is like a bridge between the two partitions which no longer exists after partitioning. It does not belong to any of the partitions. So, when a big query that is used to query this partitioned graph (many small graphs), it is most likely that our query spans across the cut-edge, thus may lose some of the results. So, handling the cut edges is very important for an effective graph partitioning. In this work reported a design on how to handle the cut edges by doing 2-hop traversal along the cut edge. Also reported three strategies for graph partitioning that would help in indexing and querying.

Graph processing means that an algorithm is executed on a graph. The vertex-centric programming model of Signal/Collect exploits the fact that most graph algorithms can be expressed in two operations on a vertex. First, a vertex signals its state to neighboring vertices along the edges. Second, a vertex collects the incoming signals and computes its new state based on them. Both the signal and the collect steps are executed repeatedly for each vertex until a termination condition is fulfilled. In a synchronous execution, all the vertices are at the same time either in the signal phase or in the collect phase. An asynchronous execution overrides this restriction and executes the signal and collects steps for each vertex independently. The Signal/Collect framework is scalable because it is able to use additional resources to speed up the execution of algorithms. Parallelism is exploited by distributing the vertices of a graph to multiple workers, which then process their assigned vertices in parallel. Distributed workers running on different compute nodes are also supported.

Multi-Level Graph Partitioning

In multi-level approaches, the coarsening phase usually consists of iterative contractions of a graph G = (V, E). A contraction of a graph merges (non-empty) subsets of V as well as their incident edges by replacing them with a new vertex/edge with a weight equal to the sum of the weights of the original vertices/edges. Each contraction operates on a higher level of the graph, beginning with the input graph, and computes the next lower level at which point the process is repeated (after clustering the new level). Figure 1 depicts a simple contraction of a small graph.

Contraction is one of the most commonly used methods for the coarsening phase. In the sequential case, many algorithms compute matchings on the graph and contract the matched edges which halves the number of vertices per contraction.

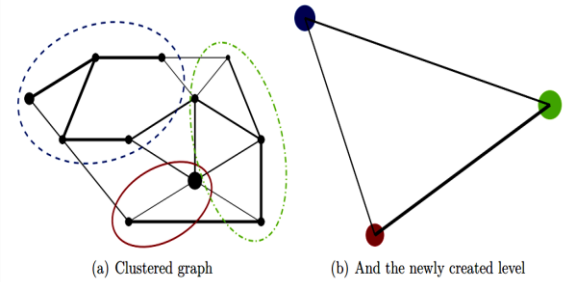

Figure 1 A small example of a contraction

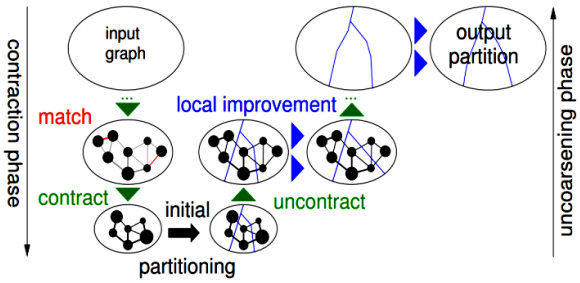

Figure 2 The general multi level graph partitioning scheme

With this definition of contraction, it is easy to see that cuts (referring to the edges inducing the cut) in a lower level of the graph have the same weights as the induced cuts in the original graph and clusters in a lower level have the same weight as the induced clusters in the original graph. This property is important to warrant the multi-level approach.

This method ensures that partitions of the coarsest level have the same global cut and imbalance as the corresponding clustering on the input graph. By contracting the graph iteratively, we get several levels of the graph. This also shrinks the graph, so we can use more expensive partitioning algorithms (which are expected to yield high-quality partitionings) on the most contracted level, use that partitioning one level above and then run the cheaper label propagation algorithm to improve the current solution.

3.Literature review

SR. NO.	TITLE	AUTHORS	YEAR	APPROACH
1	Distributed arithmetic Architectures for FIR Filters-A Comparative review	G. NagaJyothi and S. SriDevi,	2017	Finite impulse response (FIR) filter is an influential block in various signal processing applications.
2	Weighted Partitioning for Fast Multiplierless Multiple-Constant Convolution Circuit,"	G. D. Licciardo, C. Cappetta, L. Di Benedetto and M. Vigliar,	2017	A new radix-3 partitioning method of natural numbers, derived by the weight partition theory
3	Pipelined block-lifting-based embedded processor for multiplying quaternions using distributed arithmetic,	N. A. Petrovsky, A. V. Stankevich and A. A. Petrovsky,	2016	A systematic design of the of the integer-to-integer invertible quaternionic multiplier based on the block-lifting structure
4	Efficient adaptive RLFIR filter based on Distributed Arithmetic Logic using Reversible gates	K. Durga and A. Sivagami,	2016	An efficient adaptive Reversible Logic Finite Impulse Response filter (RLFIR) based on Distributed Arithmetic (DA)
5	An asynchronous double precision floating point multiplier,	S. Nair and T. S. B. Sudarshan,	2015	A higher radix multiplier design is used to form a block in floating point datapath helps in increasing the efficiency of the design.
6	Modification of the Architecture of a Distributed Arithmetic,	V. Lesnikov, T. Naumovich and A. Chastikov	2015	A way to reduce the number of steps require a distributed arithmetic.
7	Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic	S. Y. Park and P. K. Meher,	2013	A novel pipelined architecture for low-power, high-throughput, and low-area implementation of adaptive filter based on distributed arithmetic (DA)

G. NagaJyothi and S. SriDevi, [1] Finite impulse response (FIR) filter is an influential block in various signal processing applications. The complexities in VLSI implementation of FIR filters is dominated by the number of multiply and accumulate (MAC) operations. Distributed Arithmetic (DA) is an alternative technique where the MAC operations can be replaced by a series of look-up tables and addition operations. FIR filter based on DA are computationally efficient because of high degree of mechanization involved in the implementation of MAC operations using DA. Many reconfigurable and non-reconfigurable FIR filter architectures can be developed using DA. This work reviews the existing FIR filter architectures based on DA. LUT based DA and LUT-less DA are the significant methods in the implementation of non-reconfigurable filters. This brief summarizes the area and power reports of the existing non-reconfigurable FIR filter architectures based on both LUT based DA and LUT-less DA. One dimensional and two dimensional systolic DA based architectures for FIR filter implementation are also briefed. DA based adaptive FIR techniques are explained. This work presents the comparative results of FIR and adaptive FIR filter architectures in terms of area, power, area-delay product, minimum cycle period and energy per sample. This survey can form a basis for further research on DA based FIR filter architectures.

G. D. Licciardo, C. Cappetta, L. Di Benedetto and M. Vigliar, [2] A new radix-3 partitioning method of natural numbers, derived by the weight partition theory, is employed to build a multiplierless circuit that is well suited for multimedia filtering applications. The partitioning method allows

conveniently premultiplying 32-b floating-point filter coefficients with the smallest set of parts composing an unsigned integer input. In this way, similar to the distributed arithmetic, shifters and recoding circuitry, typical of other well-known multiplier circuits, are completely substituted with simplified floating-point adders. Compared to the existent literature, targeted to both field-programmable gate array and std_cell technology, the reported solution achieves state-of-the-art performances in terms of elaboration velocity, achieving a critical path delay of about 2 ns both on a Xilinx Virtex 7 and with CMOS 90-nm std_cells.

N. A. Petrovsky, A. V. Stankevich and A. A. Petrovsky, [3] This work presents a systematic design of the of the integer-to-integer invertible quaternionic multiplier based on the block-lifting structure and pipelined embedded processor of the given multiplier using distributed arithmetic (DA) as a block of M-band linear phase paraunitary filter banks (LP PUFB) based on the quaternionic algebra (Q-PUFB) for the lossy-to-lossless image coding. A bank Q-PUFB based on the DA block-lifting structure reduces the number of rounding operations and has a regular layout. Since the block-lifting structures with rounding operations can implement the integer-to-integer transform (Q-PUFB).

K. Durga and A. Sivagami, [4] This brief presents an efficient adaptive Reversible Logic Finite Impulse Response filter (RLFIR) based on Distributed Arithmetic (DA) using Reversible gates. Reversible logic is one of the most essential issues at present time due to its power reduction effectiveness in circuit designing. The delay and the logical resources of the reported design were significantly reduced by using add one carry select adder in the inner product of the adaptive filter. The existing carry save adder in the adaptive filter is replaced by the reported add one carry select adder and logic gates in add one carry select adder is replaced by reversible logic gates in order to reduce the power consumption. The logical resources and delay is reduced to half when compared to the existing carry save adder and the power consumption is reduced to half by changing reversible gates. This work presents quantum implementation and combinational circuit of all basic reversible gates and its VHDL code. All reversible logic gates are verified and simulated by Xilinx 8.2i.
S. Nair and T. S. B. Sudarshan, [5] Floating point multiplier is a key element in most digital applications such as filters, processors, embedded systems and control units. It is mainly used because of the need for a large dynamic range or in rapid prototyping applications where the actual number range has not been clearly specified. Hence increasing throughput of floating point multiplier is an issue to be taken into account. Asynchronous communication helps in increasing the throughput when the comparison is made with a synchronous system. Also usage of an higher radix multiplier design in the floating point datapath helps in increasing the efficiency of the design. Mantissa computation is the most critical part in floating point multiplication and hence efficient multiplier must be used to compute the same. A higher radix multiplier design is used to form a block in floating point datapath helps in increasing the efficiency of the design. Radix selection of the booth multiplier depends on the chosen IEEE format. For higher precision format there is a requirement for higher radix multiplier. For minimizing the disadvantages of higher radix multiplier design, an high speed adder can be used.

V. Lesnikov, T. Naumovich and A. Chastikov, [6] Distributed arithmetic has been widely used for area-time efficient implementation of inner products. It is a bit-serial computational operation that forms an inner product of a pair of vectors in a single direct step. If the word length of the vectors is large, the sequential implementation of the distributed arithmetic require a large number of steps. This work proposes a way to reduce the number of steps. It is offered to refuse the sequential analysis of bit slices. Instead, the contents of the memory are assigned sequentially combinations of bit slices.

S. Y. Park and P. K. Meher, [7] This brief presents a novel pipelined architecture for low-power, high-throughput, and low-area implementation of adaptive filter based on distributed arithmetic (DA). The throughput rate of the reported design is significantly increased by parallel lookup table (LUT) update and concurrent implementation of filtering and weight-update operations. The conventional adder-based shift accumulation for DA-based inner-product computation is replaced by conditional signed carry-save accumulation in order to reduce the sampling period and area complexity. Reduction of power consumption is achieved in the reported design by using a fast bit clock for carry-save accumulation but a much slower clock for all other operations. It involves the same number of multiplexors, smaller LUT, and nearly half the number of adders compared to the existing DA-based design. From synthesis results, it is found that the reported design consumes 13% less power and 29% less area-delay product

(ADP) over our previous DA-based adaptive filter in average for filter lengths N = 16 and 32. Compared to the best of other existing designs, our reported architecture provides 9.5 times less power and 4.6 times less ADP.

4.Problem formulation

Fast Fourier Transform (FFT) is used to build various image processing systems and application specific Digital Signal Processing (DSP) hardware. Currently almost all reported designs for FFT use ROMs or memory for complex twiddle multiplications. Proper techniques must be followed to eliminate the need of multipliers in FFT design[8-14]. One of the most frequently used and significant method to eliminate the multipliers used in FFT design is using Distributed Arithmetic for twiddle multiplications. While using DA technique, one must do precise shifting to reduce the number of adders.

5.Conclusion

In this work various related work has been revived the high performance ongoing FIR channels, which process the multiplier signals with insignificant delays (latencies). A principle plan is to understand the channels with the slightest computational load, the minimum conceivable runtimes or, at the end of the day, inside an insignificant number of processor cycles. From a hypothetical perspective, the unpredictability of algorithms can be communicated by the quantity of activities (principally number juggling tasks, similar to increases and augmentations). On functional machines be that as it may, the execution of digital signal processor (DSP) algorithms is unequivocally influence by numerous further viewpoints, similar to the memory get to, the reserving and fanning conduct and the standard allelization capacities. Keeping in mind the end goal to accomplish an ideal execution, these angles must be thought about similarly. The quick Fourier change surely turned out to be extremely prominent and presumably the most vital device in computerized signal preparing. Today, a great many people connect fast convolution with FFT-based convolution algorithms. Furthermore, there are a few decent contentions, why FFT-based convolution is a phenomenal decision.

References

[1] NagaJyothi G, SriDevi S. Distributed arithmetic architectures for FIR filters-A comparative review. In wireless communications, signal processing and networking (WiSPNET), International Conference on 2017 (pp. 2684-90). IEEE.

[2] Licciardo GD, Cappetta C, Di Benedetto L, Vigliar M. Weighted partitioning for fast multiplierless multiple-constant convolution circuit. IEEE Transactions on Circuits and Systems II: Express Briefs. 2017; 64(1):66-70.

[3] Petrovsky NA, Stankevich AV, Petrovsky AA. Pipelined block-lifting-based embedded processor for multiplying quaternions using distributed arithmetic. In Embedded Computing (MECO), 5th Mediterranean Conference on 2016 (pp. 222-25). IEEE.

[4] Durga K, Sivagami A. Efficient adaptive RLFIR filter based on distributed arithmetic logic using reversible gates. In intelligent systems and control (ISCO), 2016 10th International Conference on 2016 (pp. 1-4). IEEE.

[5] Nair S, Sudarshan TS. An asynchronous double precision floating point multiplier. In electrical, computer and communication technologies (ICECCT), IEEE International Conference on 2015 (pp. 1-5). IEEE.

[6] Lesnikov V, Naumovich T, Chastikov A. Modification of the architecture of a distributed arithmetic. In East-West Design & Test Symposium (EWDTS), 2015 IEEE 2015 (pp. 1-4). IEEE.

[7] Park SY, Meher PK. Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic. IEEE Transactions on Circuits and Systems II: Express Briefs. 2013; 60(6):346-50.

[8] Hewlitt RM, Swartzlantler ES. Canonical signed digit representation for FIR digital filters. In signal processing systems, 2000. SiPS 2000. IEEE Workshop on 2000 (pp. 416-26). IEEE.

[9] Tsoumanis K, Axelos N, Moschopoulos N, Zervakis G, Pekmestzi K. Pre-encoded multipliers based on non-redundant radix-4 signed-digit encoding. IEEE Transactions on Computers. 2016; 65(2):670-6.

[10] Peled A, Liu B. A new hardware realization of digital filters. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1974; 22(6):456-62.

[11] Voronenko Y, Püschel M. Multiplierless multiple constant multiplication. ACM Transactions on Algorithms (TALG). 2007; 3(2):11.

[12] Aksoy L, Flores P, Monteiro J. Efficient design of FIR filters using hybrid multiple constant multiplications on FPGA. In Computer Design (ICCD), 2014 32nd IEEE International Conference on 2014 (pp. 42-7). IEEE.

[13] Berkeman A, Owall V, Torkelson M. A low logic depth complex multiplier using distributed arithmetic. IEEE Journal of Solid-State Circuits. 2000; 35(4):656-9.

[14] Basiri M MA, Sk NM. An efficient hardware-based higher radix floating point MAC design. ACM Transactions On Design Automation of Electronic Systems (TODAES). 2014; 20(1):15.