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

An extensive survey on weighted partitioning for multiple-constant convolution fast multiplierless circuit

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

Abstract
The graph partitioning issue is a deliberation of this utilization case there are additionally lots of other applications of the issue. Simulations using the finite element method, forms during VLSI configuration, route planning and logical processing all prompt or advantage from solving a graph partitioning problem. Gaussian convolutions are maybe the frequently utilized a graph partitioning problem in low-level PC vision undertakings. The discrete convolution kernel is when all is said in done not equivalent to the examined rendition of the consistent convolution bit. It turns out to be the inspected adaptation of the convolution of the consistent convolution kernel and the continuous interpolation kernel. Multiplication and Fast Fourier Transform are essential devices utilized as a part of the Digital Signal Processing applications. Every one of them is register escalated segment of broadband beam forming applications, for example, those by and large utilized as a part of software characterized radio and sensor systems. These are often utilized bit tasks in signal and image processing including versatile frameworks. This work reported a broad study on different constant convolution fastmultiplierless Circuit in view of weighted partitioning algorithm.

Keywords
Constant convolution, Multiplierless circuit, Weighted partitioning.

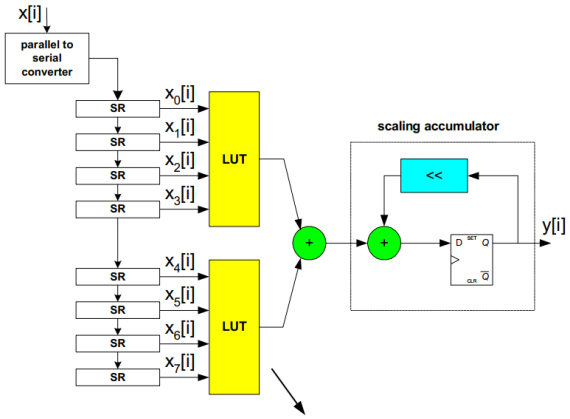
1.Introduction
Gaussian convolutions are perhaps the most often used image operators in low-level computer vision tasks. Surprisingly though, there are precious few articles that describe efficient and accurate implementations of these operators.Multiply and Accumulate (MAC) is one of the basic blocks used in many digital signal processing systems. The general structure of a MAC unit consists of a multiplier, an adder and a shifter. Elimination of multiplier in MAC unit can be made possible by using algorithms such as Distributed Arithmetic (DA).Thus; DA eliminates the use of multipliers and ROM to carry out many computations such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), etc [8].

As digital signal processing (DSP) is integrated into more devices, time to market and the ability to make late design changes becomes important. Software can give the flexibility in design, allowing late design changes but its performance is poor compared to hardware. Software executes in a sequential manner where hardware can execute in a truly parallel way. On the other hand, creating an application specific integrated circuit (ASIC) takes the longer time to make and once done it is not changeable. This is where a field programmable gate array (FPGA) becomes a great solution by combining the strengths of hardware and software.

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].

The key insight in this computation is that consists of binary constants of the form of power of 2. This allows for the precomputation of all these values, storing them in a lookup table, and using the individual inputs xi as an address into the lookup table. Here, each line calculates the final product by using one bit (of the weight) from all inputvalues. This effectively replaces the constant multiplication with a lookup table. Then the computation corresponding to each line of the is performed by addressing the lookup table with the appropriate values as dictated by the individual input variables. Each line is computed serially and the outputs are shifted by the appropriate amounts (i.e. 0, 1, 2, …, B-1 bits). Figure 1.1 presents a visual depiction of the DA version of inner product computation. The input sequence is fed into the shift register at the input

sample rate. The serial output is presented to the RAM based shift registers at the bit clock rate which is B+1 times (n is number of bits in a data input sample) the sample rate. The RAM based shift register stores the data in a particular address. The outputs of registered LUTs are added and loaded to the scaling accumulator from LSB to MSB, and the result is accumulated over time. For an n bit input, n+1 clock cycles are needed for a symmetrical filter to generate the output [10].



Address	Data
0000	0
0001	C_0
0010	C_1
…	…
1111	$C_0+C_1+C_2+C_3$

Figure 1 A DA FIR filter block diagram

In a conventional MAC, with a limited number of MAC blocks, the system sample rate decreases as the filter length increases due to the increasing bit width of the adders and multipliers and consequently the increasing critical path delay. However, this is not the case with serial DA architectures since the filter sample rate is decoupled from the filter length. As the filter length is increased, the throughput is maintained but more logic resources are required. While the serial DA architecture is efficient by construction, its performance is limited by the fact that the next input sample can be processed only after every bit of the current input sample is processed. Each bit of the current input sample takes one clock cycle to process.

2.Graph partitioning

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 workreported 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 [9].

The structure of a graph might change over time by adding or removing vertices and edges. They are also dynamic in the sense that during an execution, certain edges might not transport any signals whereas others might be active in each step. Furthermore, these messaging patterns might change over time. Most algorithms for classical graph partitioning do not deal with such dynamic graphs. Third, the primary goal of graph partitioning in the context of Signal/Collect is not to find a perfect partitioning, which might take a lot of time, but to improve the runtime performance of the system. From this discussion, it can be concluded that most of the classical graph partitioning algorithms are not appropriate to solve the present problem. Figure 2.1 shows that in graph processing systems like

Signal/Collect, the partitioning of the graph can happen before, during, or after the computation. Based on these phases, differentiate between static and dynamic graph partitioning.This terminology is also used [1].

Static graph partitioning is executed before a computation is started or after it is ended. Partitioning the graph after a computation is ended is reasonable in case the graph is executed again with the same or a different algorithm. Static partitioning is a pre or postprocessing step to the execution of a graph and does not modify the partitions during the computation. It is straightforward to integrate into the system, because not many changes to the core of Signal/Collect are necessary. Furthermore, numerous methods and established tools are available to statically partition a graph.

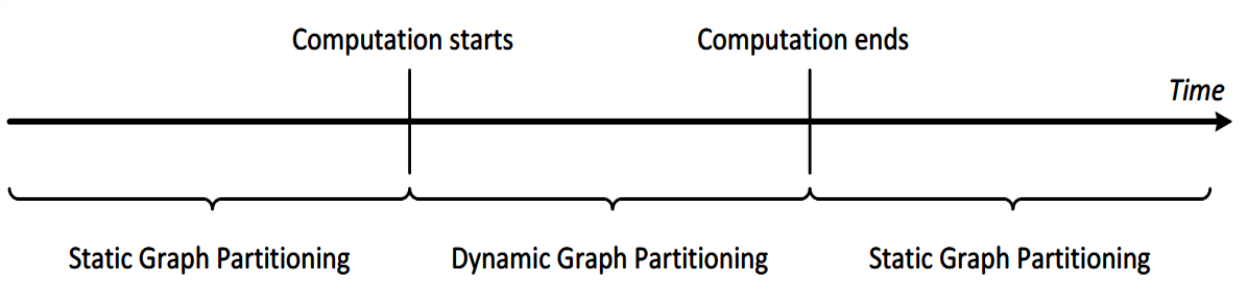


Figure 2 Static and dynamic graph partitioning.

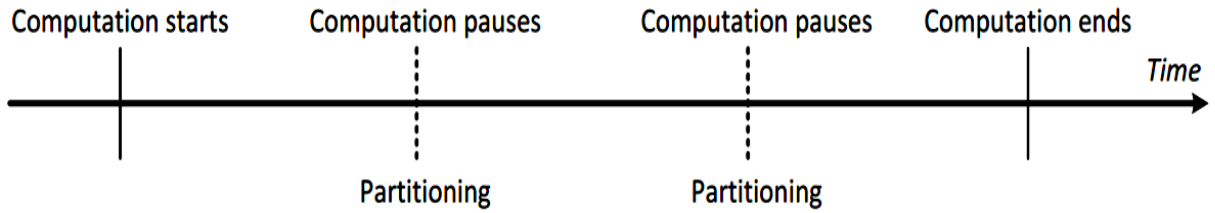


Figure 3 Serial dynamic graph partitioning

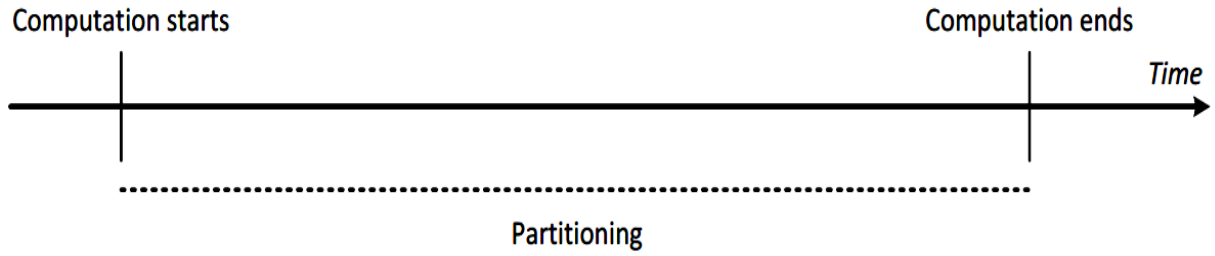


Figure 4 Parallel dynamic graph partitioning.

The preeminent advantage of this approach is that it is able to adjust the partitions continuously during the computation. Compared to the static approach, it is, however, more difficult to integrate into the system as it affects the core of the Signal/Collect system and cannot be implemented as a pre- or post processing step to the execution of the graph.Dynamic graph partitioning can be further categorized into serial and parallel partitioning.

Figure 2 shows that in case of serial dynamic graph partitioning, the computation is paused to partition the graph. Parallel dynamic graph partitioning partitions the graph while the computation is running, as Figure 2.3 shows.

3.Literature review

Table 1 Refrences of the Paper

SR. NO.	Title	Authors	Year	Approach
1	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
2	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.
3	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)
4	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
5	Modification of the Architecture of a Distributed Arithmetic,	V. Lesnikov, T. Naumovich and A. Chastikov	2015	A way to reduce the number of stepsrequire adistributed arithmetic
6	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.
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. D. Licciardo, C. Cappetta, L. Di Benedetto and M. Vigliar, [1]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.

G. NagaJyothi and S. SriDevi, [2] 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.

K. Durga and A. Sivagami, [3] 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.

N. A. Petrovsky, A. V. Stankevich and A. A. Petrovsky, [4]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).

V. Lesnikov, T. Naumovich and A. Chastikov, [5]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. Nair and T. S. B. Sudarshan, [6] 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.

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. One of the most frequently used and significant method to eliminate the multipliers used in FFT design is usingDistributed 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 ASICs and capabilities of the DSP processors in the computationally intensive applications such as digital signal processing.Technological developments have caused acceleration in the development of the area of DSP. One of them is the devising of an efficient algorithm to calculate the Discrete Fourier Transform. In addition they offer the flexibility in hardware and shorter time to market. In the meanwhile, multiplierless circuit hardware design flows is a challenging task due to the integration of several design tools and specific architecture that imposes design challenges to the designers.The major objective of this work is to review approaches to reduce the computation in designing a circuit and hence the hardware resources required for that by devising a novel method called Distributed arithmetic method.

References

[1] 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.

[2] 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.

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

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

[5] 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.

[6] Nair S, Sudarshan TS. An asynchronous double precision floating point multiplier. In Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on 2015 (pp. 1-5). 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, SiPS. 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), 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.