



OPTIMIZING FINITE IMPULSE RESPONSE FILTERS FOR ULTRA-FAST SIGNAL PROCESSING THROUGH CRITICAL PATH MINIMIZATION

Naguru Ayesha, Chamarthi Ravi

Electronics and communication Engineering,
MJR College of Engineering and Technology, Piler, Andhra Pradesh, India-517214

Abstract : This study has been undertaken to investigate the determinants of stock returns in Karachi Stock Exchange (KSE) using two assets pricing models the classical Capital Asset In this project, we propose a new hardware architecture of a very high-speed finite impulse response (FIR) filter using fine-grained seamless pipelining. The proposed full-parallel pipeline FIR filter can produce an output sample in a few gate delays by placing the pipeline registers not only in between components, but also across the components. A precise critical path analysis at the gate level allows to create an appropriate pipelining strategy depending on the throughput requirement. This paper also presents two alternative architectures, each offering different trade-offs in terms of area and throughput rate. The proposed FIR filters are synthesized to measure the maximum throughput and the balance between complexity and speed. The effectiveness of the proposed method is synthesized and simulated using Xilinx ISE.

I. INTRODUCTION

A digital multiplier is the fundamental component in general-purpose microprocessors and in digital signal processors. It is very easy to find such operation in signal processing algorithms and matrix arithmetic. Typically, multipliers are implemented using the modified Booth algorithm (MBA), which requires modest hardware but results in unacceptably long delays. Lately, there were efforts to speed up MBA multipliers by using parallel implementations. We were successful in developing high-speed multiply-accumulate structures based on the Baugh-Wooley algorithm (BWA), and applying these structures to several digital filtering applications. Partial-Product Generation: This can be achieved using several techniques such as the BWA, the Booth algorithm (BA), or the MBA. For an -bit multiplier, the number of summands is for BWA, for BA, and for MBA. In addition to the encoding step, the BA and MBA algorithms also require generation of the two's complement of the multiplier which introduces extra delay. The delay for two's-complement generation is not trivial, but has been consistently neglected in most of the proposed designs in the literature. Partial-Product Addition: This is done using a carry ripple adder for serialparallel multipliers. For parallel multipliers, the addition is accomplished using carry-save techniques, Wallace trees, or summand skip. However, the last two techniques require irregular wiring and extra hardware. In this paper, we employ the more regular neighbor-to-neighbor interconnection pattern while achieving the same desired result. Final Adder: When the number of partial products is reduced to sum and carry words, a final adder is required to generate the multiplication result. The number of bits of the final adder is the sum of the number of bits of the multiplier and multiplicand. Thus, the data path width is usually doubled and the delay of this stage is most severe. Traditionally, authors simply state that 4-bit CLAs can be used. In this paper, we use a mix of 2- and 4-bit CLAs to reduce the delay and area requirements.

II. EXISTING METHOD

In the existing method, FIR filter with modified booth encoders for multiplication is performed and the accumulation of products for each tap is performed and the final product is obtained. A filter design can be defined as, it is the process of choosing the length and coefficients of the filter.

The intention is to set the parameters so that the required parameters like a stop band and pass band will give the result from running the filter. A K-tap FIR filter produces the output sample $y(n)$ using the K most recent input samples of $x(n)$, $x(n-1), \dots$

, $x(n-K+1)$ as $y(n) = \sum_{k=0}^{K-1} w(k)x(n-k)$, (1) where $w(k)$ for $0 \leq k \leq K-1$ represents the $(k+1)$ -th FIR filter coefficient.

The K-tap FIR filter needs to perform K multiplications and $(K-1)$ additions to obtain one output sample $y(n)$. Note that the maximum throughput can be achieved when the K multipliers and the $(K-1)$ adders are used in parallel as shown in Fig below.

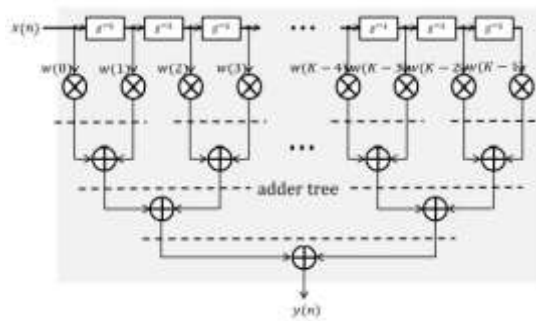


Fig: Full parallel FIR filter

Finite impulse response (FIR) filters are the most popular type of filters implemented in software. This introduction will help you understand them both on a theoretical and a practical level. Filters are signal conditioners. Each functions by accepting an input signal, blocking pre-specified frequency components, and passing the original signal minus those components to the output.

Modified Booth Algorithm Encoder:

This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit.

The SMFF performs K-tap FIR filtering through K recursive MAC computations over K clock cycles. Therefore, the area of the SMFF can be significantly reduced at the expense of throughput. The detail structure of the proposed SMFF with K 16-bit inputs and K 16-bit coefficients.

The MBE receives one input and one coefficient every clock cycle and then produces m/2 PPs. The compressor array reduces m/2 + 2 PPs, i.e., the m/2 PPs from MBE and the other 2 PPs from the registers, to 2 PPs. The generated 2 PPs are stored in the registers and taken by the compressor array again in the next cycle. Although the first stage WRT can be reused as a compressor array, one of the various compressors found in the literature can also be employed for the purpose.

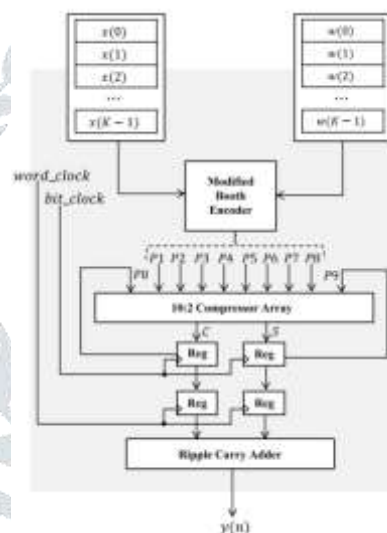


Fig : single MAC FIR filter (SMFF)

A folded FIR filter (FDFF) uses two or more MBEs and compressor arrays, but less than the number of taps of the FIR filter. The FDFF is slower than FPFF but occupies less area, and in contrast, has larger area than the SMFF but runs faster. That is, an optimal trade-off between SMFF and FPFF can be found by adjusting the number of MAC units according to the required timing constraint. The structure of the proposed FDFF, especially with two MAC units. Two pairs of inputs and coefficients are fed into MBE every Tbit-clock.

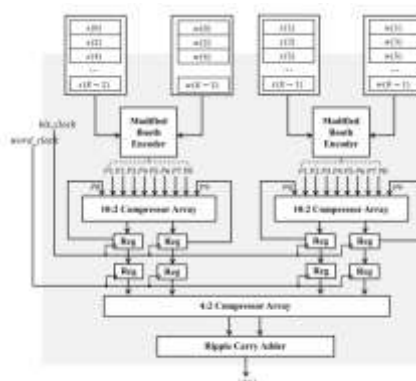


Fig : folded FIR filter (FDFF)

If the input and coefficients are m-bits each, then each MBE produces m/2 PPs, which are used as inputs to the m/2 + 2:2 compressor array. If the bit-widths of input and coefficient are 16-bits, two 10:2 compressor arrays, each of which is the same as the one in the SMFF, and one 4:2 compressor array are employed. Each 10:2 compressor array produces two

PPs, i.e., a total of 4 PPs, which are stored in the upper 4 registers, then fed back to the compressor arrays with the next MBE output on the next bit-clock cycle. If the number of taps of the FIR filter is K where K is even, this process is repeated $K/2$ times to have one output sample. After $K/2$ repetitions, the 4 PPs accumulated in the registers are reduced to 2 PPs using a 4:2 compressor array, and finally the output of the filter is obtained using the RCA. Since the operations of the 4:2 compressor array and RCA only need to be performed once per $K/2$ bit-clock cycles, Tword-clock can be set to have a $K/2$ times longer period than the Tbit-clock. Also, note that Tbit-clock can be set to equal to the one of the SMFF, that is, TFF + TMBE + 6TXOR because their critical paths are the same.

III. PROPOSED SYSTEM:

In the proposed method, to derive a very high speed FPFF through finegrained seamless pipelining has been implemented. The throughput of the FIR filter can be improved by inserting pipeline registers into the critical path at the expense of increasing latency and area. Therefore, the design of the pipeline FIR filter should begin with finding the critical path that causes the longest propagation delay.

There are a lot of paths from each input bit to each output bit, of which the critical path with the longest propagation delay should be found through analysis. However, it is not possible to predict which gates in the standard cell library will be selected during the synthesis process. Also, the delay for each gate is not fixed either as it can change depending on constraints imposed on the synthesis process as well as temperature, load delay factor, output load capacitance, etc.

All the design procedures are based on precise critical path analysis. The first possible location of the pipeline registers is indicated by a dotted line before level 1, which stores the results of the hidden MBE calculation. Then, the delay of the first pipeline stage becomes 6DG, that is, to obtain $\alpha 0$ in the first PP. Next, for pipelining of the WRT, the registers can be inserted before and after the WRTs, as well as between levels inside WRTs.

The pipeline registers can also be inserted between the FAs inside the RCA. The possible locations of the pipeline registers, where the top dotted line is between the last level of WRT and RCA. Note that inserting a pipeline register after every individual FA results in a 4DG delay, and it does not help to decrease the delay of whole FIR filter since MBE's delay is 6DG.

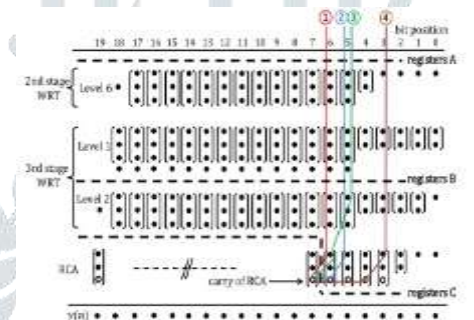


Fig : Propagation delay in pipeline stage containing WRT and RCA.

Let us locate the critical path of Type-5 first in order to formulate its propagation delay. Note that the critical path can be formed in one of two paths in WRT area, that is, one is from A to S of the FA in the same bit position, that is, in the vertical down direction, and the other is from A to COUT in the diagonal down direction. On the other hand, in RCA area, we know that the critical path goes from a lower bit position to higher bit positions (from right to left) via carry propagation. Now, let us see how the critical path is formed in Type-5 with the example.

The last level-6 and the third stage WRT are combined with RCA in order to constitute Type-5. First let us consider only the area higher than the fifth bit position having the regular dot diagram, where both WRT and RCA are filled with only FAs.

Modified Full Adder:

Where this modified full adder consists of two 4:1 multiplexer. Using this modified full adder gets decreases with low power consumption.

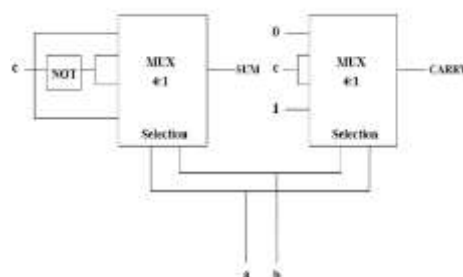


Fig: Proposed full adder multiplexers

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, ' n ' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines. Since there are ' n ' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as Mux. 4x1 Multiplexer has four data inputs I3, I2, I1 & I0, two selection lines s1 & s0 and one output Y. The block diagram of 4x1 Multiplexer is shown in the following figure. In this modified full adder consists of two 4:1 multiplexers here a, b are two selection lines for both the multiplexers C is the only input to the multiplexer that gives sum as output, 0,c,1 are the inputs for the multiplexer that gives carry as the output. After studying the conventional full adder design, a multiplexer based design is modeled by identifying its operation using truth table and the input and output values when logic 0 and logic 1 as input at the consequent input line. Using this multiplexer based adder designs it

can be observed that there is a maximum reduction in the gate count of the logic design, thus making it an efficient implementation.

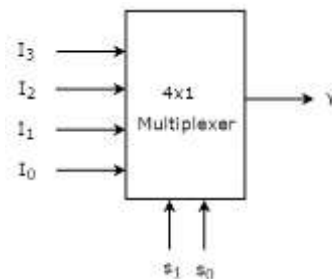


Fig: Multiplexer

IV. VLSI DESIGN FLOW

The VLSI design cycle starts with a formal specification of a VLSI chip, follows a series of steps, and eventually produces a packaged chip.

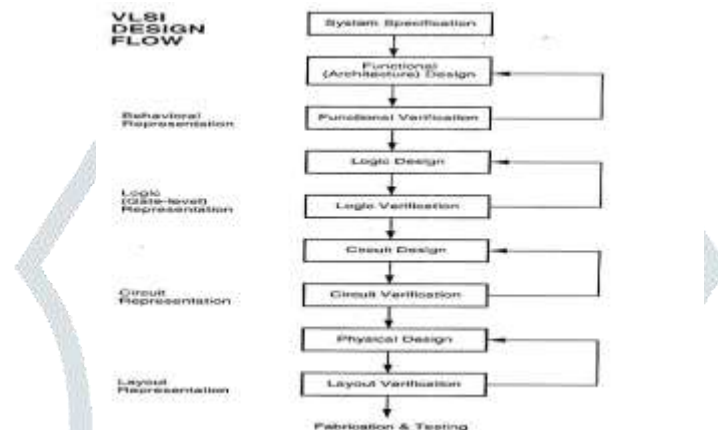


Fig : VLSI Design Flow

System Specification:

The first step of any design process is to lay down the specifications of the system. System specification is a high level representation of the system. The factors to be considered in this process include: performance, functionality, and physical dimensions like size of the chip. The specification of a system is a compromise between market requirements, technology and economical viability. The end results are specifications for the size, speed, power, and functionality of the VLSI system.

Architectural Design

The basic architecture of the system is designed in this step. This includes, such decisions as RISC (Reduced Instruction Set Computer) versus CISC (Complex

Instruction Set Computer), number of ALUs, Floating Point units, number and structure of pipelines, and size of caches among others. The outcome of architectural design is a Micro-Architectural Specification (MAS).

Behavioral or Functional Design:

In this step, main functional units of the system are identified. This also identifies the interconnect requirements between the units. The area, power, and other parameters of each unit are estimated.

Modules. The key idea is to specify behavior, in terms of input, output and timing of each unit, without specifying its internal structure.

The outcome of functional design is usually a timing diagram or other relationships between units.

Logic Design:

In this step the control flow, word widths, register allocation, arithmetic operations, and logic operations of the design that represent the functional design are derived and tested.

This description is called Register Transfer Level (RTL) description. RTL is expressed in a Hardware Description Language (HDL), such as VHDL or Verilog.

This description can be used in simulation and verification

Circuit Design:

The purpose of circuit design is to develop a circuit representation based on the logic design. The Boolean expressions are converted into a circuit representation by taking into consideration the speed and power requirements of the original design. Circuit Simulation is used to verify the correctness and timing of each component

The circuit design is usually expressed in a detailed circuit diagram. This diagram shows the circuit elements (cells, macros, gates, transistors) and interconnection between these elements. This representation is also called a netlist.

And each stage verification of logic is done.

MODULE:

A module is the basic building block in Verilog. It can be an element or a collection of low level design blocks. Typically, elements are grouped into modules to provide common functionality used in places of the design through its port interfaces, but hides the internal implementation.

Syntax:

```
module<module name> (<module_port_list>);
```

```
.....
```

```
<module internals> //contents of the module
```

....

Endmodule

V. XILINX VERILOG HDL TUTORIAL

Getting started

First we need to download and install Xilinx and ModelSim. These tools both have free student versions. Please accomplish Appendix B, C, and D in that order before continuing with this tutorial. Additionally if you wish to purchase your own Spartan3 board, you can do so at Digilent's Website. Digilent offers academic pricing. Please note that you must download and install Digilent Adept software. The software contains the drivers for the board that you need and also provides the interface to program the board.

Introduction

Xilinx Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The design procedure consists of (a) design entry, (b) synthesis and implementation of the design, (c) functional simulation and (d) testing and verification. Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description language (HDL) – Verilog or VHDL or a combination of both. In this lab we will only use the design flow that involves the use of Verilog HDL.

The CAD tools enable you to design combinational and sequential circuits starting with Verilog HDL design specifications. The steps of this design procedure are listed below:

Create Verilog design input file(s) using template driven editor.

Compile and implement the Verilog design file(s).

Create the test-vectors and simulate the design (functional simulation) without using a PLD (FPGA or CPLD).

Assign input/output pins to implement the design on a target device.

Download bitstream to an FPGA or CPLD device.

Test design on FPGA/CPLD device

A Verilog input file in the Xilinx software environment consists of the following segments:

Header: module name, list of input and output ports.

Declarations: input and output ports, registers and wires.

Logic Descriptions: equations, state machines and logic functions.

End: endmodule

Opening Designs:

Use the Flow Navigator or Flow menu to select the following commands:

Open Elaborated Design

Open Synthesized Design

Open Implemented Design

The Flow > Open Implemented Design command populates the Vivado IDE as shown in below figure.

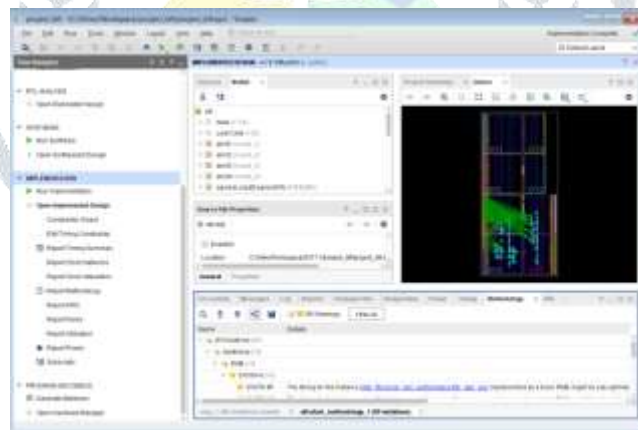


Fig: Implemented design

VI. XILINX VIVADO SIMULATION PROCEDURE:

After completion of synthesis we will go simulation in order to verify the functionality of the implemented design.

Click on Run Simulation and set the module that is need to Run

Next double click on Run Behavioral Simulation to check the errors. If no errors are found then double click on simulate behavioral model to get the output waveforms.

After clicking on simulate behavioral model, the simulation widow will appear pass the input values by making force constant and if it is clock by making force clock. Mention the simulation period and run for certain time and results will appear as shown in following window. Verify the results to the given input values.

Using the Schematic Window:

You can generate a Schematic window for any level of the logical or physical hierarchy. You can select a logic element in an open window, such as a primitive or net in the Netlist window, and use the Schematic command in the popup menu to create a Schematic window for the selected object.

An elaborated design always opens with a Schematic window of the top-level of the design, as shown in below figure.



As synthesis and implementation complete, DRC violations, timing values, utilization percentages, and power estimates are also populated. To open the Project Summary, do either of the following:

- [illegible]

Fig: Project Summary

RTL schematic: The simulation window will appear the input values by using the Modified booth in the partial stage while the pipeline is accrossing the filter.



Technology Schematic: By using synthesis process we can find the technology schematic ,it has the lot of pipelining is acrossing the filter,by using the enlarge option we can find the exact values of the output.

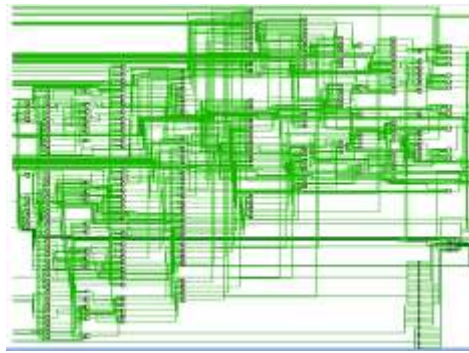


Fig :Output for Technology Schematic

Simulation results: The simulation window will appear pass the input values by making force constant and if it is clock by making force clock. Mention the simulation period and run for certain time and results will appear as shown in following window. Verify the results to the given input values.

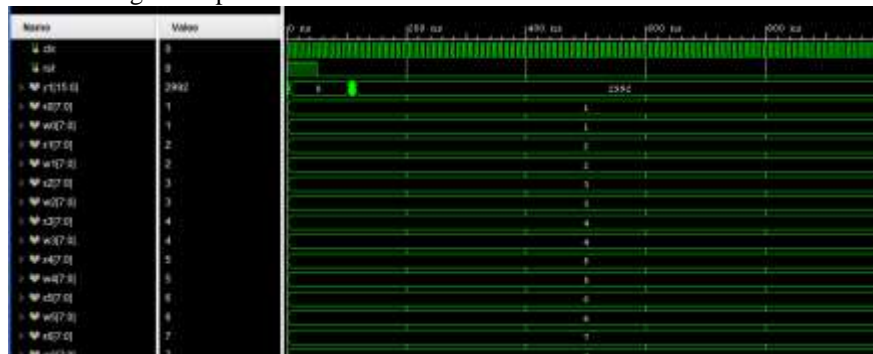


Fig :Result of Simulation

Area: To find the area in the simulation process we have to choose the report utilization option in which the SLICE LUTs is the total area.

Resource	Utilization	Available	Utilization %
LUT	1351	20800	6.50
FF	2064	41600	4.96
IO	274	106	258.49

Power:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power: 56.62 W (Junction temp exceeded!)
Design Power Budget: Not Specified
Power Budget Margin: N/A

Delay: We can find the delay time by report timing summary ,go to the unconstrained path in which select the setup option. By selecting setup option we can get the total delay time.

```

Max Delay Paths:
-----
Slack: inf
Source: pp4_r_reg[4]/C
        (rising edge-triggered cell FDRE)
Destination: pol_r_reg[9]/D
        (none)
Path Group:
Path Type: Max at Slow Process Corner
Data Path Delay: 5.653ns (Logic 1.247ns (22.059%), route 4.406ns (77.941%))
Logic Levels: 6 (FDRE=1, LUT5=1, LUT6=1, LUT6=3)
  
```

Evaluation table for Area, Delay, Power:

	Area(LUT's)	Delay(ns)	Power(W)
FIR	1608	5.853	63.688
Extension FIR	1351	5.653	56.62

VII. CONCLUSION

In this paper, we are implementing pipeline FIR filters that can provide very high throughput. The proposed design begins with a reference full-parallel design based on MBE, hierarchical Wallace tree network, and RCA. The propagation delay of the FIR filter has been approximated in terms of unit gate delay through precise analysis, which helped to establish an efficient pipelining strategy at gate level. As a result, the proposed full-parallel design can achieve very high throughput through fine-grained seamless pipelining. In this paper, we have also proposed alternative structures, which provides relatively high throughput while significantly reducing the area. The significance of this paper is that the proposed FIR filter can provide scalability to DSP applications that require very high throughput rate.

VIII. REFERENCES

- [1] A. Eghbali, H. Johansson, O. Gustafsson, and S. J. Savory, "Optimal leastsquares FIR digital filters for compensation of chromatic dispersion in digital coherent optical receivers," *J. Lightw. Technol.*, vol. 32, no. 8, pp. 1449–1456, Apr. 2014.
- [2] D. Kang, Y. Kang, and Y. Hong, "VLSI implementation of fractional motion estimation interpolation for high efficiency video coding," *Electron. Lett.*, vol. 51, no. 15, pp. 1163–1165, Jul. 2015.
- [3] M. S. Hosseini and K. N. Plataniotis, "High-accuracy total variation with application to compressed video sensing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3869–3884, Sep. 2014.
- [4] J. Wang, J. Lin, and Z. Wang, "Efficient hardware architectures for deep convolutional neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 6, pp. 1941–1953, Jun. 2018.
- [5] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, "An architecture to accelerate convolution in deep neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1349–1362, Apr. 2018.
- [6] P. Meher and S. Park, "Design of cascaded CORDIC based on precise analysis of critical path," *Electronics*, vol. 8, no. 4, p. 382, Mar. 2019.
- [7] P. K. Meher and M. Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*
- [8] P. K. Meher, "Seamless pipelining of DSP circuits," *Circuits, Syst., Signal Process.*, vol. 35, no. 4, pp. 1147–1162, Apr. 2016.
- [9] S. K. Patel and S. K. Singhal, "Area-delay and energy efficient multioperand binary tree adder," *IET Circuits, Devices Syst.*, vol. 14, no. 5, pp. 586–593, Aug. 2020.
- [10] L. Dadda and V. Piuri, "Pipelined adders," *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 348–356, Mar. 1996.
- [11] S. A. Khan, *Digital Design of Signal Processing Systems: A Practical Approach*. Hoboken, NJ, USA: Wiley, 2011.
- [12] S.-R. Kuang, J.-P. Wang, and C.-Y. Guo, "Modified booth multipliers with a regular partial product array," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 5, pp. 404–408, May 2009.
- [13] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1253–1258, Oct. 2006.
- [14] T. V. Fontanari, G. Paim, L. M. G. Rocha, P. Ucker, E. Costa, and S. Bampi, "An efficient N-bit 8-2 adder compressor with a constant internal carry propagation delay," in *Proc. IEEE 11th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2020.
- [15] A. Fathi, B. Mashoufi, and S. Azizian, "Very fast, high-performance 5- 2 and 7-2 compressors in CMOS process for rapid parallel accumulations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 6, pp. 1403–1412, Jun. 2020.
- [16] T.-B. Juang, P. K. Meher, and K.-S. Jan, "High-performance logarithmic converters using novel two-region bit-level manipulation schemes," in *Proc. Int. Symp. VLSI Design, Autom. Test*, Apr. 2011, pp. 1–4.
- [17] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.
- [18] P. K. Meher and S. Y. Park, "CORDIC designs for fixed angle of rotation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 2, pp. 217–228, Feb. 2013.
- [19] P. K. Meher and S. Y. Park, "Reconfigurable FIR filter for dynamic variation of filter order and filter coefficients," *J. Semicond. Technol. Sci.*, vol. 16, no. 3, pp. 261–273, Jun. 2016.
- [20] C. Cheng and K. K. Parhi, "Hardware efficient fast parallel FIR filter structures based on iterated short convolution," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 8, pp. 1492–1500, Aug. 2004.
- [21] Y.-T. Hwang and C.-L. Su, "Parallel and pipelined architecture designs for distributed arithmetic-based recursive digital filters," in *Proc. 9th IEEE Workshop VLSI Signal Process.*, Oct. 1996, pp. 35–44.
- [22] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 6, pp. 346–350, Jun. 2013.